AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING
Department of Communications and Networking (Comnet)

Abdulfetah Khalid

Load Balancing of Elastic Data Traffic in Heterogeneous Wireless Networks

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, January 23, 2013

Supervisor: Samuli Aalto, Professor, PhD

Instructor: Pasi Lassila, D.Sc

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author:	Abdulfetah Khalid
Name of the Thesis:	Load Balancing of Elastic Data Traffic in Heterogeneous Wireless Networks
Date:	January 23, 2013 **Number of pages:** 94
Department:	Department of Communications and Networking
Professorship:	S-38
Supervisor:	Samuli Aalto, Professor
Instructor:	Pasi Lassila, D.Sc

The increasing amount of mobile data traffic has resulted in an architectural innovation in cellular networks through the introducion of heterogeneous networks. In heterogeneous networks, the deployment of macrocells is accompanied by the use of low power pico and femtocells (referred to as microcells) in hot spot areas inside the macrocell which increase the data rate per unit area.

The purpose of this thesis is to study the load balancing problem of elastic data traffic in heterogeneous wireless networks. These networks consist of different types of cells with different characteristics. Individual cells are modelled as an $M/G/1-PS$ queueing system. This results in a multi-server queueing model consisting of a single macrocell with multiple microcells within the area. Both static and dynamic load balancing schemes are developed to balance the data flows between the macrocell and microcells so that the mean flow-level delay is minimized. Both analytical and numerical methods are used for static policies. For dynamic policies, the performance is evaluated by simulations.

The results of the study indicate that all dynamic policies can significantly improve the flow-level delay performance in the system under consideration compared to the optimal static policy. The results also indicate that MJSQ and MP are best policies although MJSQ needs less state information. The performance gain of most of the dynamic polices is insensitive with respect to the flow size distribution. In addition, many interesting tests are conducted such as the effect of increasing the number of microcells and the impact of service rate difference between macrocell and microcells.

Keywords: Heterogeneous wireless networks, load balancing, multi-server queueing model, $M/G/1-PS$, LTE

Acknowledgements

This work was carried out at the Department of Communications and Networking at Aalto University as a part of the HEWINETS project funded by TEKES, Ericsson, and Cassidian Systems. There are people to whom I own a debt of gratitude for making the course of my research both an enriching and an enjoyable experience.

First and foremost, I wish to express my sincerest thanks to my supervisor, Professor Samuli Aalto, for the enthusiastic guidance and encouragement he has given me throughout the thesis work.

I am also honoured to have been able to work with my instructor, D.Sc. Pasi Lassila. His advice and assistance has been appreciated. I hope to continue working with him in the future. I would also like to thank all my colleagues from the COMNET Department for providing me a prosperous and well spirited working environment.

Finally, I would like to thank my friends and family, especially my parents who have given me a lifetime of loving support and encouragement. Henceforth, I am dedicating this thesis work to them.

Otaniemi, January 23, 2013

Khalid Abdulfetah Hadi

Contents

List of Abbreviations

3GPP	The 3rd Generation Partnership Project
3GPP2	The 3rd Generation Partnership Project 2
CN	Core Network
CDMA	Code Division Multiple Access
EGPRS	Enhanced GPRS
FTP	File Transfer Protocol
FIFO	First In First Out
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HSUPA	High Speed Uplink Packet Access
IID	Independently and Identically Distributed
JSQ	Join the Shortest Queue
LFF	Least Flow-time First
LTE	Long-Term Evolution
LWL	Least Work Load

MIMO	Multiple Input Multiple Output
MJSQ	Modified Join the Shortest Queue
MME	Mobility Management Entity
MP	Myopic Policy
MSC	Mobile services Switching Centre
OFDMA	Orthogonal Frequency-Division Multiplexing
P-GW	Packet Data Network Gateway
PS	Processor Sharing
RAN	Radio Access Network
RR	Long Term Evolution
RTT	Round Trip Time
S-GW	Serving Gateway
SC-FDMA	Single Carrier Frequency Division Multiple Access
SGSN	Serving GPRS Support Node
SITA	Size Interval Task Assignment
SITA-E	Size Interval Task Assignment with Equal Load)
SITA-V	Size Interval Task Assignment with Variable Load
SMS	Orthogonal Frequency Division Multiplexing
SNIR	Signal to Noise and Interference Ratio
SRPT	Shortest Remaining Processing Time
TAGS	Task Assignment based on Guess
TCP	Transmission Control Protocol

TDMA	Time Division Multiple Access
TL	Transport Layer
UDP	User Datagram Protocol
UE	User Equipment
USIM	UMTS Subscriber Identity Module
UMTS	Universal Mobile Telecommunications System
VLR	Visitor Location Register
WCDMA	Wideband Code Division Multiple Access

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Mobile data communications has undergone significant evolution in recent years. The introduction of High Speed Downlink Packet Access (HSDPA) enabled mobile broadband internet for the first time. This resulted in an exponential increase in the traffic volume of the mobile data. In the future, it is expected that there will be more mobile data demand than there has been before. According to Nokia Siemens Networks and Ericsson, it is estimated that the data usage rate will increase 1000 percent from 2010 to 2020 [1]. The main reason for this is the increasing level of penetration of data-intensive devices, such as smart phones, and an increasing level of usage per device.

The rapidly increasing mobile data traffic has resulted in a major challenge for the operators. Data volumes are growing at a rate that exceeds the operators' ability to increase capacity. Capacity growth typically comes from growth in the number of sites, increased spectrum resources and enhancement of radio technologies. The huge gap between the data demand growth and network capacity growth demanded further innovations. This resulted in the concept of heterogeneous networks in which some of the data traffic is offloaded onto other smaller networks using microcells, picocells, femtocells and Wi-Fi access points. It is expected that the deployment of heterogeneous networks fulfills

the expected future data demands.

In a cellular network, efficient allocation of resources (channels) to each cell is needed due to limited bandwidth. This problem becomes even worse when some cells in the system are congested while others are not. Therefore, this causes a hotspot problem in which the quality of service in congested cells is degraded by a considerable amount. It is well known that the hot spot problem can be solved by dynamically balancing the load of the hot spot cells in cellular networks, i.e., by dispatching the excess traffic of the highly loaded cells to the less loaded cells in the system. To overcome the hotspot problem, the allocation of the users to the base stations should be guided by the load balancing principle. By using the load balancing principle, it is possible to divide workload over the base stations as evenly as possible, thus fulfilling the target of minimizing the mean delay in the system.

1.2 Research problem

This thesis addresses the load balancing problem for heterogeneous wireless networks. These networks contain different types of cells in which each cell may have different characteristics. In general, these cells are divided into macrocells and microcells. The macrocells are similar to the conventional base stations that we use in today's networks. They provide the basic coverage to the whole cell area. The microcells are low power base stations which are required to cover areas that a macrocell cannot cover efficiently. The microcells that we consider in this thesis are assumed to have a wired backhaul connection to the internet. There are different kinds of microcells such as picocells, relay nodes and femtocells which are going to be deployed based on the situations and locations needed.

This thesis studies how load balancing of elastic traffic between a macrocell and a number of microcells is realised. Elastic traffic consists of adaptive TCP flows such as the transfer of digital documents. The basic approach is to model individual cells, using the concepts of flow level modelling, with $M/G/1-PS$ queueing system. Since there are different types of cells, we have a multi server

case with different arrival and service rates for each cell. The main task of this thesis is to develop static and dynamic load balancing schemes that balance the data flows between microcells and the macrocell. In the static case, the load balancing problem is formulated as an optimization problem in which the mean flow delay is minimized. For dynamic load balancing schemes, a simulator is needed to study their performance.

In general, the system is modelled to consist of a single macro cell and a number of micro cells. Two cases are considered with respect to the characteristics of the servers and the users. In the so-called symmetric case, it is assumed that the arrival rate and service rate of different microcells, which are within a single macrocell, become identical. For the second case, called the asymmetric case, the traffic parameters are different for different microcells. One task in this thesis is, for both cases, to find analytical results for the optimal static policy. The main task is to compare the performance of dynamic policies to the optimal static policy with respect to the mean number of flows.

1.3 Outline of the thesis

The thesis is organized into eight chapters. Chapter 2 introduces the technological background information of LTE and heterogeneous networks. The theoretical background that covers some relevant concepts related to the thesis topic is presented in Chapter 3. In Chapter 4, we introduce flow level modelling of elastic data traffic.

In Chapter 5, we introduce distributed server systems and discuss several dispatching policies proposed for distributed systems. The chapter is concluded with a discussion of known results of dispatching policies.

The research problem is formulated as a load balancing problem in heterogeneous networks in Chapter 6. The system model and both static and dynamic policies are proposed in this chapter.

Chapter 7 presents numerical results for the policies proposed in Chapter 6. Several traffic and system scenarios are introduced to study the performance

of the dynamic load balancing policies. Both analytical and simulation results are given for these traffic scenarios. Finally, in the last chapter, conclusions are drawn and future works for this research problem are also proposed.

Chapter 2

LTE and heterogeneous networks

In this chapter, we will go through the basics of LTE and heterogeneous networks. In the first section, 2.1, the evolution of mobile communication networks from the first generation to the coming fourth generation will be introduced. Technological background of LTE and heterogeneous networks will be discussed in more detail in Sections 2.2 and 2.3, respectively. Finally, in Section 2.4, the traffic offload in heterogeneous networks is discussed. The book by Holma and Toskala [1] has been used as the main source for this chapter.

2.1 Cellular networks before LTE

It has been a norm to divide mobile communication technologies into generations. The first-generation of wireless telephone technology (1G) being the analog mobile radio systems of the 1980s. It was replaced by 2G which is the first digital mobile systems. The 2G cellular networks were commercially launched based on the GSM standard. This standard used Time-Division Multiple Access (TDMA). The second-generation digital mobile communications brought the opportunity to provide data services over the mobile-communication networks. The primary data services introduced in 2G were

text messaging (Short Message Services, SMS) and circuit-switched data services. They enabled e-mail and other data applications with peak data rate of 9.6 kbit/s. The demand of higher data rates resulted in the development of GPRS which enables packet data over cellular systems. Due to its enhanced connection speed, GPRS was regarded as 2.5G. GPRS offered a maximum data rate of 115 kbit/s. Enhanced GPRS (EGPRS) which gives a further increase in the data rate of packet data communication was developed and introduced in 1999 .

There have been two major kinds of 3G standards called UMTS and CDMA2000. The first one was standardized by 3rd Generation Partnership Project (3GPP) using Wideband Code Division Multiple Access (WCDMA) radio interface, while CDMA2000 was standardized by 3GPP2. There have been enhancements in data rates after WCDMA, which is known as Release 99. Release 99 supports circuit-switched voice and video services, and data services over both packet-switched and circuit switched bearers. The first major addition of radio access features to WCDMA was HSPA, which was added in Release 5 with High Speed Downlink Packet Access (HSDPA) and in Release 6 with Enhanced Uplink called High Speed Uplink Packet Access (HSUPA). HSPA has been further enhanced in Release 7 (known as HSPA+) with higher-order modulation and, for the first time in a cellular communication system, multi-stream 'MIMO' operation (Multiple-Input Multiple-Output antenna system). Further enhancements of HSPA+ have been introduced in Release 8 in parallel to the first release of LTE which is a standard for wireless communication of high-speed data for mobile phones and data terminals. Figure 2.1 shows the maximum achievable data rates that have been specified by 3GPP, which is currently the dominant standards development group for mobile radio systems within the 3GPP evolution track.

The evolution track shown in Figure 2.2 is also developed in the 3GPP. The figure shows the publication date of the standardization on one side and the deployment date on the other. The multiple access technologies used in the third generation UMTS family is WCDMA. LTE has adopted Orthogonal Frequency-Division Multiplexing (OFDM), which is the access technology dominating the latest evolutions of all mobile radio standards. In continu-

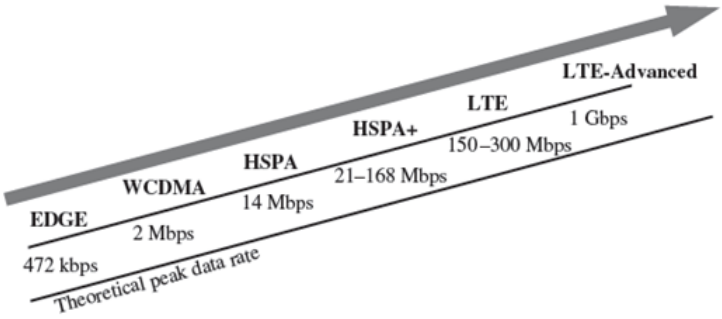

Figure 2.1: Peak data rate and evolution of 3GPP technologies [1]

ing the technology progression from the GSM and UMTS technology families within 3GPP, the LTE system can be seen as completing the trend of expansion of service provision beyond voice calls towards a multiservice air interface.

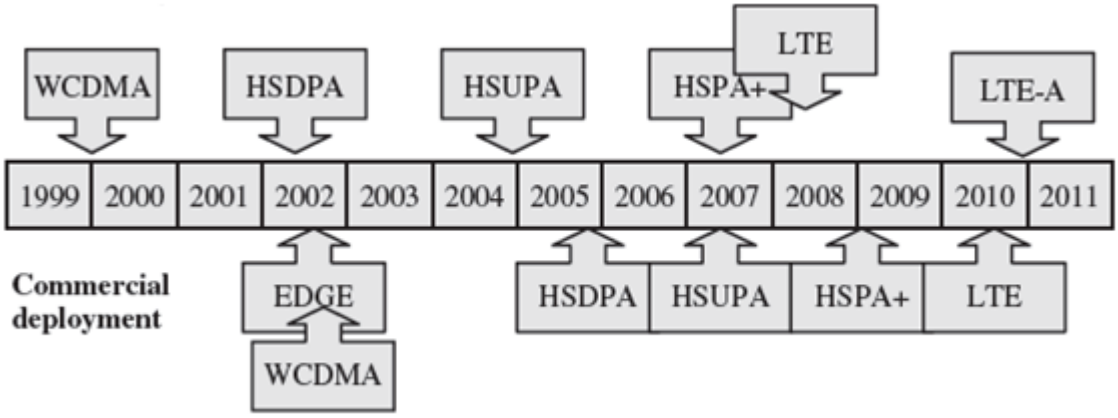

Figure 2.2: Schedule of 3GPP standard and commercial deployment [1]

2.1.1 WCDMA (UMTS)

WCDMA (Wideband Code Division Multiplex Access) is the most commonly used UMTS air interface standard found in 3G mobile telecommunications networks. The UMTS system utilizes a similar architecture that has been used in the second generation systems. The UMTS system consists of a number of logical network elements that each have a defined functionality. The network elements in the UMTS are grouped into three.

- Radio Access Network (RAN/UTRAN) that is responsible for handling radio-related functionalities. It contains Node B and Radio Network Controller (RNC). The basic tasks done by the Node B are coding, interleaving, rate adaptation, modulation and spreading. RNC is responsible for controlling radio resources of Node B's in its operation area.

- Core Network (CN), which is responsible for switching and routing calls and data connections to external networks. The basic CN architecture for UMTS is based on the GSM network with GPRS. It is divided into circuit switched and packet switched domains. Some of the circuit switched elements are Mobile services Switching Centre (MSC), Visitor location register (VLR) and Gateway MSC. Packet switched elements are Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN).

- User Equipment (UE) which is any device used directly by an end-user to communicate to the network. It communicates with several NodeBs. UE contains Mobile equipment (ME) and UMTS Subscriber Identity Module (USIM).

2.1.2 HSPA

HSPA is a combination of two mobile telephony protocols called High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA). HSPA is an extension of WCDMA that improves the overall performance of the WCDMA technology by increasing peak data rates and system capacity. It boosts the peak data rates up to 14 Mbit/s in the downlink and 5.76 Mbit/s in the uplink. In 2008, Evolved HSPA (also known as HSPA+) was released which provides data rates up to 84 Mbit/s in the downlink and 22 Mbit/s in the uplink (per 5 MHz carrier) with multiple input, multiple output (MIMO) technologies.

2.2 LTE

The introduction of bandwidth consuming smart phones has resulted in a dramatic increase in the data demand from the mobile users. So the demand for new services and for higher peak bit rates and system capacity are the main reasons for the evolution of the mobile technology to 4G. There are also other driving forces behind LTE development such as wire line capability evolution, need for lower cost wireless data delivery and competition of other wireless technologies [2].

Several targets were set for LTE Release 8 in order to fulfill the driving factors for its development. Some of the basic targets are

- achieving high peak data rates up to 50 Mbps in the uplink and 100 Mbps in the downlink,
- improving the spectral efficiency with the use of OFDM in downlink, SC-FDMA in uplink and up to 4x4 MIMO technology and
- providing very low latency by reducing the time required for setup, transfer delays and handover delays.

Besides these performance targets, there were other targets such as ensuring high level of mobility and security, allocating frequencies from 1.5 MHz up to 20 MHz which is required for backward compatibility with earlier 3GPP releases.

In general, the main LTE target was to improve the network scalability and to minimize the end-to-end latency by reducing the number of network elements. The general reason to start architecture evolution was the drive towards flat Packet Switched optimized networks. Figure 2.3 shows the basic architecture of Release 8. The overall network structure consists of two parts called eNodeB and the Core Network (CN). The eNodeB inherits all algorithms that are located in RNC in WCDMA/HSPA architecture. It has evolved from NodeB of WCDMA and HSPA in a way that it will perform the functionality of both NodeB and Radio Network Controller (RNC) of previous technologies.

All radio protocols, (part of) mobility management, header compression and packet retransmissions are located in eNodeB. The core network consists of the Mobility Management Entity (MME), Serving Gateway (S-GW) and Packet Data Network Gateway (P-GW). The main functionalities of these are

- MME - It is responsible for keeping track of the location of all UEs in its service area. It also requests the appropriate resources from eNodeB and S-GW which it selects for the UE.
- P-GW - It is responsible for IP address allocation for the UE, as well as QoS enforcement and flow-based charging.
- S-GW - All user IP packets are transferred through the Serving Gateway, which serves as the local mobility anchor for the data bearers when the UE moves between eNodeBs. It also retains the information about the bearers when the UE is in the idle state and temporarily buffers downlink data while the MME initiates paging of the UE to re-establish the bearers.

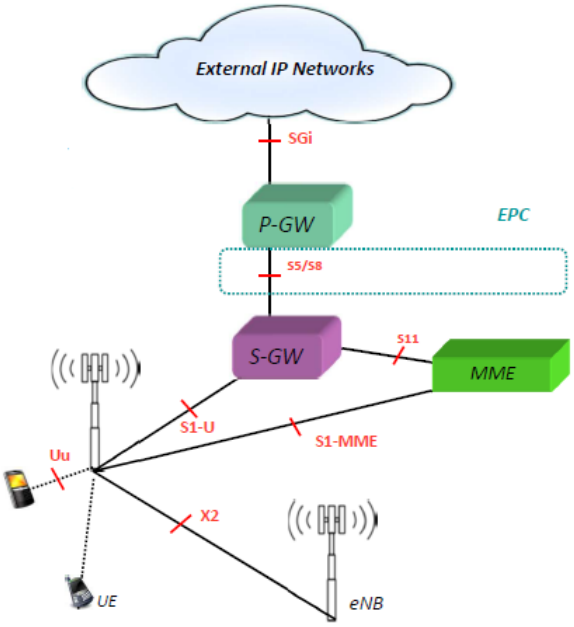

Figure 2.3: LTE overall architecture [2]

2.3 Heterogeneous networks

It has been witnessed in the past that wireless data traffic is increasing exponentially. This is because of the increasing penetration of data intensive devices, such as smart phones, tablets and the like, and an increasing level of usage per device. So there is a need to increase the capacity of current mobile networks. This can be realized in two ways.

The first one is to enhance the current radio links. In LTE Release 10, different enhancements are made to increase the radio link capacity by introducing enhanced MIMO with up to 8x8 antenna configuration, by introducing a bandwidth of up to 100MHz using carrier aggregation, and by using higher modulation schemes. It has been shown in [4] that operators can expect their network capacities to increase around 29% per year. However, according to a Cisco prediction, the demand of the data rates is growing at 108% per year. This means there is a significant gap where the capacity enhancement techniques mentioned above can not fulfill the demand. In addition, the conventional cellular architecture is designed to serve a wide range of areas. User achieved data rate varies across the cell as the user moves far from the base station as a result of inter-cell interference and the low transmit power constraint of the user equipments. To address these issues operators needed to come up with another plan.

The second option to fulfill the growing data demand is to consider architectural innovations. This can be realised by introducing *heterogeneous networks*. They are network architectures with small cells (microcells, picocells and femtocells) overlaying the macrocell network. The deployment of heterogeneous networks supports macros, picos, femtos and relays in the same spectrum, which will utilize a diverse set of base-stations that can be deployed to improve spectral efficiency per unit area. This results in the traditional definition of a cell to be changed in such a way that the number of 'cells' is increased and the cost per unit of capacity is decreased significantly. By using a mix of macro, pico, femto and relay base-stations, heterogeneous networks enable flexible and low-cost deployments and provide a uniform broadband experience to users anywhere in the network.

Heterogeneous network, which avoids the drawbacks of homogeneous networks and is expected to satisfy the currently required data demands, is a network deployment strategy which creates a multi-layer cellular network as depicted in Figure 2.4. In heterogeneous networks, the deployment of macro eNodeBs, which provides the basic coverage, is accompanied by the deployment of low power nodes such as pico cells, relay nodes, remote radio heads, and femto cells. Deploying a pico cell or relay nodes in cell edges helps cell edge users to experience similar performance as those users close to the macro eNodeB by increasing the SINR level. By deploying femto cells inside buildings, coverage holes can be avoided. Moreover, the capacity demand in hot spot areas can be fulfilled by deploying many low power nodes. Furthermore, relay nodes and remote radio heads can be used to serve remote locations, and at the same time to reduce the cost for backhaul connections and operational expenditure, as they use a wireless backhaul connection. Therefore, by using heterogeneous networks, all users throughout the network can fully utilize the radio link capacity and the 'data rate per unit area' can be significantly increased.

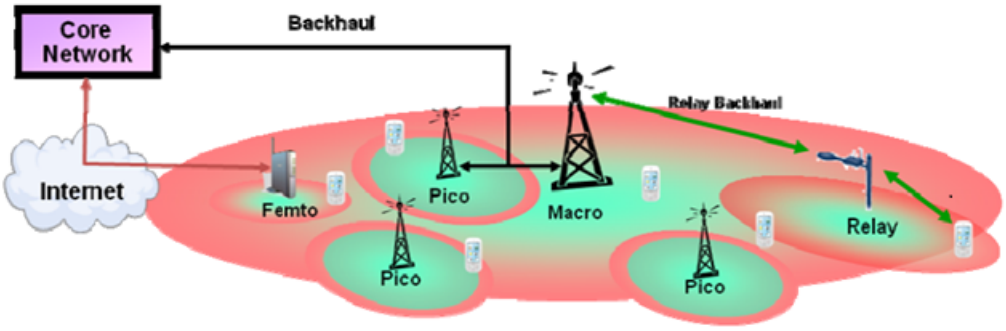

Figure 2.4: Heterogeneous network utilizing a mix of macro, pico, femto and relay base-stations [2]

2.3.1 Component technologies of heterogeneous networks

Macrocells

Macrocells are conventional base stations that use dedicated backhaul and are open to public access. The transmit power is around 43 dBm and its

antenna gain is in between 12-15 dBi. The macrocell network is deployed by the operator based on detailed network planning and link budget calculations.

Picocells

Picocells are low power base stations that use dedicated backhaul connections and are open to public access. Their typical transmit power ranges from 23 dBm-30 dBm and its antenna gain is in between 0-5 dBi.

Relays

Relays are base stations using the same spectrum as backhaul and access. They use a similar power as picocells. Relaying is one of the key features of LTE-Advanced, where a relay node receives a signal from user equipment and transfers it to the donor eNodeB, and vice versa [5]. Relays and picocells are very similar to macrocells with the exception that they have a smaller size and lower power. In addition, they are usually deployed to cover a smaller area and serve users in dense clusters since they are relatively easy to deploy.

Femtocells

Femtocells are consumer deployable base stations that utilize the broadband connection of the consumer as the backhaul. They are suitable for indoor locations such as homes and offices (enterprise). Their transmit power is limited to 23dBm. Therefore, a single femtocell can serve only a few users. However, a network of femtos can be used to cover larger areas such as an enterprise. An enterprise or public access femtocell has the functional equivalency of a picocell.

2.4 Traffic offload in heterogeneous networks

In a traditional cellular network, all of the traffic to and from mobile phones and mobile internet devices travel from the device to a cell site that is typ-

ically a fraction of mile away. However, in heterogeneous networks either a microcell or a femtocell carries the traffic from the phone to the operator or another internet destination. The effect of traffic offload is that it results in a significant reduction of total traffic travelling over wide area radio networks of the operators. In addition, it also benefits consumers as it offers a higher bandwidth for mobile data connection. In this section, we will give an overview of traffic offload in heterogeneous networks. In Section 2.4.1 and 2.4.2, traffic offload using microcells and femtocells are introduced respectively.

2.4.1 Traffic offload via microcells and picocells

Traffic offload via micro/picocells can be used for outdoor offload scenarios. Microcell and picocell users will generally experience higher data rates. The most significant advantages of using a microcell come from the fact that it allows co-channel deployments, two or more cells in a cellular mobile radio system that use the same frequency, and has limited interference impact on macro users [3]. In addition, these cells use the existing backhaul and network architecture. Figure 2.5 illustrates how a microcell or picocell can be deployed in the same frequency channel as the macrocell. Common use cases for micro and picocells are:

- in situations where macrocell requires a coverage extension;
- for data capacity supplements in hotspot areas;
- for extending indoor coverage.

Simulation results presented in [3] show that there is a significant improvement in total cell throughput with the use of microcells to offload the traffic. They found that for every additional microcell introduced in the macrocell based network, the troughput of the cell increased by 100%.

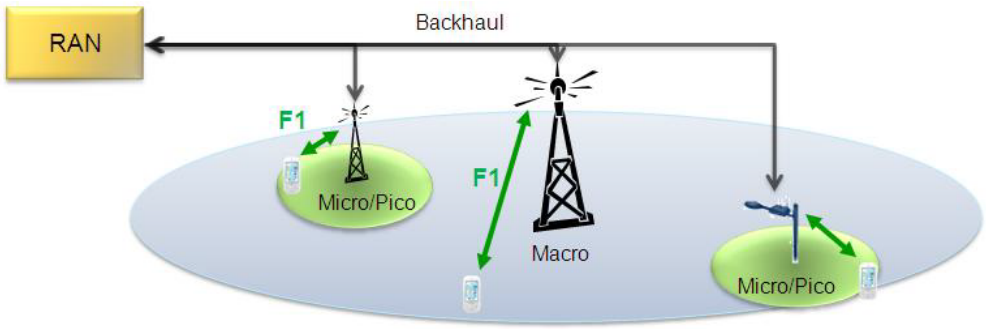

Figure 2.5: Illustration of a deployment of micro/pico cell in the same frequency channel as the macro [3]

2.4.2 Traffic offload via femtocells

Femtocells are deployed at a wireless subscriber's home and the consumer attaches it to his or her broadband connection. Traffic then flows over the air to the femtocell and then over the internet to the operator's core network, and/or to other internet destinations [4]. One of the key advantages of using a femtocell is that the traffic generated by femtocells will not be carried by the macrocells. As a result, users in the macrocell have a better experience because of reduction in the number of users who compete for resources. Thus, using femtocells gives a dual benefit by reducing the traffic in the macro network and providing a strong signal to indoor users. Figure 2.6 shows the use case scenario of internet traffic offload using a femtocell.

The benefit of traffic offload with respect to throughput is demonstrated in [3] with simulations. The results showed that the user throughput improves significantly with the deployment of femtocells.

Although introducing femtocells can give a significant improvement on user experience it has its own set of challenges. The major challenge comes from unplanned deployment of femtocells. This results in interference between femtocells and interference between macrocell and femtocells on both uplink and downlink. So, there is a need for efficient interference management techniques in heterogeneous networks.

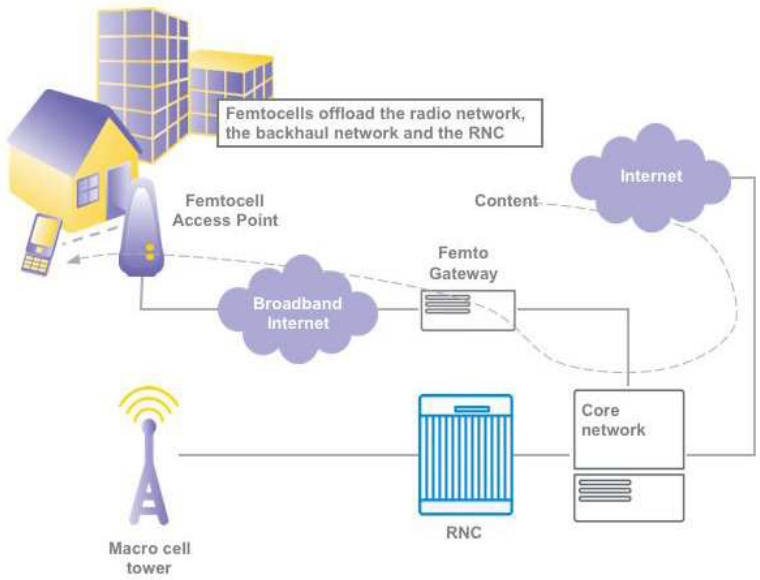

Figure 2.6: Use case scenario of internet traffic offload via a femtocell [4]

Chapter 3

Theoretical background

In this chapter, we discuss some basic concepts of queueing theory that we will need later on. We start from the introduction of heavy tailed distributions which are commonly used to model the job size distribution of the current applications. In Section 3.2, the so-called Kendall's notation is introduced. The basic scheduling disciplines are reviewed in Section 3.3. Section 3.4 introduces the Poisson process. Markov processes and birth-death processes will be presented in Section 3.5 and 3.6, respectively. In Sections 3.7 and 3.8, we will briefly describe a few basic queueing models that will be used later for modelling the problem of our interest. Most material used in this chapter is taken from the book by Kleinrock [5].

3.1 Heavy tailed distributions

The most common choice for telecommunication network design and performance analysis was based on the exponential assumption where jobs or sessions arrive as a Poisson process and they have exponential holding times. This has been successful in analysing the performance of traditional voice networks which consists of only voice traffic. However, this assumption fails in describing today's traffic where we consider packet switched networks. This is because today's applications generate traffic that is highly variable. The traffic measurements of these applications have shown that there is a signifi-

cant traffic variance over a wide range of time scales. A class of distributions that is often used to capture the characteristics of a highly variable stochastic process is called heavy tailed distributions. Heavy-tailed workloads have been found to exist in a number of modern computing environments. Researchers [6] found that heavy-tailed distributions appear to fit many recent measurements of computing systems, including file requests by users, files transmitted via the network, transmission duration's of files and files stored on servers. In addition, the observed heavy-tailed workloads include the size of files stored in Unix file systems [7], and the Unix process CPU requirements measured at UC Berkeley [8]. Many of these application environments show a mixture of task sizes spanning many orders of magnitude. In such environments, there are typically many small tasks and fewer large tasks.

Heavy-tailed properties

A heavy-tailed distribution is one whose tail decays like a power-law, that is,

$$P\{X > x\} \sim x^{-\alpha}, \text{ for } 0 < \alpha \leq 2 \tag{3.1}$$

The α parameter describes the variation of the distribution. The lower the parameter α the more variable the distribution, and the more pronounced is the heavy-tailed property, i.e., the smaller the fraction of large tasks that comprise half the load.

A set of job sizes following a heavy-tailed distribution has the following properties:

- Decreasing failure rate for the tail: In particular, the longer a task has run, the longer it is expected to continue running.
- Infinite variance (and infinite mean if $\alpha \leq 1$).
- The property that a very small fraction ($< 1\%$) of the largest workloads make up a large fraction (half) of the load. This is the most important property of the heavy-tailed distributions [9].

The simplest heavy-tailed distribution is the Pareto distribution with probability mass function

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad x \geq k, \tag{3.2}$$

where $k > 0$ is a scale parameter which specifies the minimum possible value of the job size and $\alpha > 0$ is the shape parameter. The cumulative distribution function of a Pareto random variable with parameters k and α is

$$F(x) = P\{X \leq x\} = 1 - \left(\frac{k}{x}\right)^{\alpha}, \qquad x \geq k. \tag{3.3}$$

The Pareto distribution can be used to model traffic that consists of heavy-tailed job sizes. But in practice there is some upper bound for the maximum size of an arriving job since all incoming jobs to the server have finite lengths. So Pareto distribution should be truncated with some maximum job size value. A bounded Pareto distribution is therefore used, which has a lower and upper limit on the job size distribution. Bounded Pareto is characterized by three parameters

α: the exponent of the power law which determines shape,

k: the shortest possible job size,

p: the largest possible job size.

The probability density function for the bounded Pareto distribution is:

$$f(x) = \frac{\alpha k^{\alpha}}{1 - (\frac{k}{p})^{\alpha}} x^{-\alpha-1}, \qquad k \leq x \leq p. \tag{3.4}$$

3.2 Kendall's notation

Kendall's notation is a standard notation used to describe the characteristics of a queueing system. This was proposed by D.G. Kendall [10]. Queueing

systems are discribed with the notation

$$A/B/C/D/E \tag{3.5}$$

where

- A stands for the description of the arrival process. It refers to the distribution of the arrivals of jobs into a system. The time distance between two arrivals is called interarrival time. There are several possible assumptions for the distributions of the interarrival times of arriving jobs. Typical assumptions are those where interarrival times are independently and identically distributed (IID). When the inter-arrival times are IID and exponentially distributed, the arrival process is specified as a Poisson process and denoted with M. However, when the exact distribution is not specified, it is described as general and denoted with G.
- B stands for the service time distribution. The symbols used for the probability distribution for the service time are, D for deterministic, M for exponential and G for general when the exact distribution is not specified.
- C stands for the number of servers in the system. It describes the number of entities that provide service to jobs in the queueing system.
- D stands for the capacity of the queue or buffer size. It specifies the maximum number of jobs in the system (including those in service and those who are waiting in the queue). If this number is not specified, the capacity is assumed to be infinite.
- E stands for the queueing discipline. They are also often referred to as scheduling disciplines (described in detail in the next section).

3.3 Scheduling disciplines

The scheduling discipline defines the service order of jobs in the system. It specifies whether the jobs in the system are served one-by-one or simultane-

ously. If the jobs are served one-by-one, the scheduling discipline tells

- in which order they are taken into the service, and
- whether it is possible to stop the service before completion.

Scheduling discipline categories

- A scheduling discipline is **non-preemptive** if jobs are served one-by-one until completion. In this discipline once the task is in service, it cannot be interrupted by an incoming job or the existing jobs in the queue. On the contrary, a queueing discipline is called **preemptive** if the job in the service can be interrupted by an incoming job or the existing jobs in the queue.
- A scheduling discipline is **work-conserving** if jobs are served whenever the system is non-empty. In this case, the capacity of the server is not wasted in a way that no server is idle if there is any waiting job in the system.
- A scheduling discipline is **non-anticipating** if the service decisions are based on only the history of the system.

It is possible to change the behaviour of the system considerably by changing the scheduling discipline. In addition, the choice of scheduling policy can have a significant effect on the mean performance metrics of tasks, as well as the variance and consequently our confidence in these metrics [5]. This section reviews three common scheduling policies.

First In First Out (FIFO)

FIFO serves the arriving jobs to completion in their arriving order. It is also known as First Come First Serve (FCFS). The jobs in the system are served one-by-one until completion. When a server becomes free, the first arrived job is taken into service. The system always serves the job that has been waiting

for the longest time. Therefore, it is a fair scheduling policy, in the sense that a task is rewarded for arriving earlier than another task by being serviced before it. It is a non-preemptive, work-conserving, and non-anticipating discipline.

Processor Sharing (PS)

The Processor Sharing scheduling discipline is an ideal form of the Round Robin (RR) discipline where jobs take turns to get served for a pre-fixed quantum. If the quantum is much smaller than the total service time of a specific job, the RR discipline corresponds to the PS discipline. In this case jobs are served simultaneously as soon as they enter the system. If there are n jobs in the system, each of them obtains the fraction $1/n$ of the capacity of the server.

Processor Sharing queues have a number of desirable properties. The first one is its fairness. It is a fair policy since it shares the service capacity equally among all jobs in the queue and all jobs are served simultaneously. This property solves many of the issues associated with FCFS queueing. The most important property is that a larger job cannot block smaller jobs behind it in the queue for lengthy periods of time. In addition, for an $M/G/1 - PS$ queue the mean time the job spends in the system is insensitive to job size distribution. It depends only on the mean of the job size distribution, and not on other higher moments. Thus, it is a robust policy as compared to FIFO. PS is a preemptive, work-conserving, and non-anticipating discipline.

Shortest Remaining Processing Time (SRPT)

The Shortest Remaining Processing Time scheduling discipline services the job with the least remaining service time first. Under this policy jobs are served one-by-one but not necessarily until completion. When a new job arrives with a shorter service time than the remaining service time of the job in service, the current service is stopped and the new job is taken into service. Therefore it is a pre-emptive discipline. SRPT is an anticipating discipline as it requires that the remaining service time of each job in the queue in advance. It has been shown to be optimal in minimising the mean response time in the $M/G/1$

queue [11].

3.4 Poisson process

The Poisson process is one of the most important models used in teletraffic theory. The arrival process of calls or packets can be described as a Poisson process when calls or packets are generated from a large number of independent users. A Poisson process can be defined in three different ways, where all three definitions are equivalent:

1. A Poisson process is a pure birth process, to be discussed in next section. In an infinitely short time period, dt, only one event can occur. This happens with the probability λdt and independently of previous events.

2. A Poisson process is a point process where the interarrival times are IID and follow the exponential distribution.

3. A counter process $(A(t)|t \geq 0)$ is a Poisson process with intensity λ if its increments are independent and follow a Poisson distribution:

$$A(t+\theta) - A(t) \sim \text{Poisson}\left(\lambda\theta\right).$$

The Poisson process has several interesting and useful properties. An important property is that Poisson Arrivals See Time Averages (PASTA). The PASTA property is one of the central tools in queueing theory. This property means that the distribution of the system at a randomly chosen time moment will be the same as the system state distribution an arrival observes. In other words, arriving jobs see the system as if they came into the system at a random instant of time.

Another important property is the random splitting property. It says that if a Poisson process with intensity λ is split into two processes with probabilities p_1 and p_2, such that $p_1 + p_2 = 1$ then the resulting processes are independent Poisson processes with intensities $p_1\lambda$ and $p_2\lambda$. This result can be generalized to any number of processes.

3.5 Markov processes

In this section, continuous-time Markov chains (CTMC) will be reviewed. We will introduce the basic properties of Markov processes since we will use a Markovian model in this study.

Let $X(t)$ be a discrete-state, continuous-time stochastic process with state space S. It is said to be a *Markov process* if

$$\begin{aligned} P[X(t_{n+1}) &= x_{n+1} \mid X(t_1) = x_1, ..., X(t_n) = x_n] = \\ P[X(t_{n+1}) &= x_{n+1} \mid X(t_n) = x_n] \end{aligned}$$

for all $n \in N$, $t_1 < .... < t_{n+1}$ and $x_1, ..., x_{n+1}$.

This means that future states are independent of the past and depend only on the present. So the current state contains all the required information as regards the future of the process. This results in the important Markov property where the time in a state has a memoryless (exponential) distribution. Markov processes are commonly used in modelling parts of communications networks and in the performance analysis of such parts.

A Markov process $X(t)$ is *time-homogeneous* if

$$P\{X(t+h) = y \mid X(t) = x\} = P\{X(h) = y \mid X(0) = x\}$$

for all t, $h \geq 0$ and the states $x, y \in S$. This means the probabilities $P\{X(t+h) = y \mid X(t) = x\}$ are independent of t and the conditional probability depends only on the difference of times, h. This property allows us to define state transition rates q_{ij} that are themselves independent of the time instant t.

The state transition rates q_{ij}, where $i, j \in S$, are defined as

$$q_{ij} = \lim_{h \to \infty} \frac{1}{h} P\{X(h) = j \mid X(0) = i\}.$$

If the Markov process is in state i, the conditional probability that it transfers to state j during a short time interval h is $q_{ij} + o(h)$. Let q_i denote the total

transition rate out of the state i:

$$q_i := \sum_{i \neq j} q_{ij}. \tag{3.6}$$

In addition to the state transition rates, it is useful to know the probabilities of the states. This is the probability which specifies the fraction of time the process spends in state i, i.e, $\pi = \{\pi_i : \pi_i \geq 0, i \in S\}$. We call the set of such probabilities as the equilibrium distribution.

The normalization condition says that the sum of state probabilities must sum up to one,

$$\sum_{i \in S} \pi_i = 1. \tag{3.7}$$

The global balance equations define the dependencies between the state in and out transitions. It specifies that for each state i, there are as many exits from state j as there are entries to it. It can be written as

$$\sum_i \pi_j q_{j,i} \quad = \quad \sum_i \pi_i q_{i,j}. \tag{3.8}$$

If the normalization condition and the global balance equations are satisfied, we achieve the equilibrium distribution of the Markov process.

3.6 Birth-death processes

A continuous-time and discrete-state Markov process with state space $S = \{0, 1, ...\}$ is called a *birth-death process* if state transitions are possible only between neighbouring states. A process that was in state i can change only to state $i + 1$ or $i - 1$. When a birth occurs, the process goes from state i to state $i + 1$. Similarly, when death occurs, the process goes from state i to state $i - 1$. It is assumed that the birth and death events are independent of each other. The process is specified by birth rates $\{\lambda_i\}$ and death rates $\{\mu_i\}$. A birth-death process is irreducible if and only if $\lambda_i > 0$ and $\mu_{i+1} > 0$ for

all $i \in S$. Figure 3.1 shows the state transition diagram of an infinite state irreducible birth-death process.

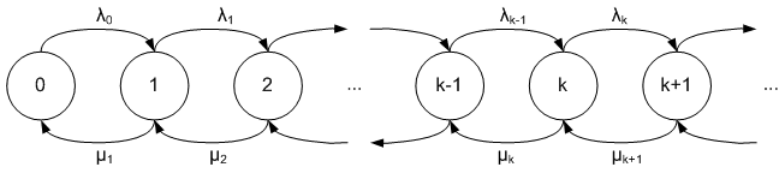

Figure 3.1: State transition diagram of an infinite state irreducible birth-death process

.

The equilibrium distribution, if it exists, can be derived using the local balance equations (LBE) and the recursive formula

$$\pi_i \lambda_i = \pi_{i+1} \mu_{i+1} \quad \Rightarrow \pi_{i+1} = \frac{\lambda_i}{\mu_{i+1}} \pi_i \tag{3.9}$$

In addition, the normalization condition can be expressed as

$$\sum_{i=0}^{\infty} \pi_i \;\; = \;\; \pi_0 \sum_{i \in S} \prod_{j=0}^{i-1} \frac{\lambda_j}{\mu_{j+1}} = 1. \tag{3.10}$$

The equilibrium distribution exists if and only if

$$\sum_{i \in S} \prod_{j=0}^{i-1} \frac{\lambda_j}{\mu_{j+1}} < \infty. \tag{3.11}$$

By means of the recursion, the equilibrium distribution can be expressed in terms of that of steady state probability 0, π_0, as

$$\pi_i = \left(\prod_{j=0}^{i-1} \frac{\lambda_j}{\mu_{j+1}} \right) \pi_0, \qquad \text{where} \quad \pi_0 = \frac{1}{1 + \sum_{i=1}^{\infty} \prod_{j=0}^{i-1} \frac{\lambda_j}{\mu_{j+1}}} \tag{3.12}$$

Birth-death processes are important because they are usually used to model queueing systems. In addition, the equilibrium distribution and related performance metrics are easy to calculate for birth-death processes.

3.7 M/M/1 queue

In this section, we will look at the properties of the $M/M/1$ queue. The $M/M/1$ queue is characterized as follows:

- Inter-arrival times are independently and identically distributed (IID) according to an exponential distribution with mean $1/\lambda$, where λ is the arrival rate. It means the arrival process is Poisson with intensity λ.
- Service times are IID and exponentially distributed with mean $1/\mu$.
- There is a single server in the system ($C = 1$).
- Infinite number of job places ($D = \infty$).

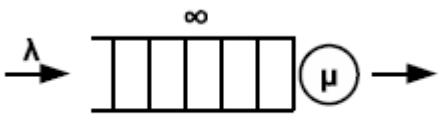

Figure 3.2: M/M/1 queueing system

Figure 3.2, shows the M/M/1 queueing system. For an inter-arrival time following an exponential distribution with the mean value $1/\lambda$ and exponentially distributed service demand with the service rate μ, we can define the traffic load of the system as $\rho = \lambda/\mu$. From the state transition diagram given in

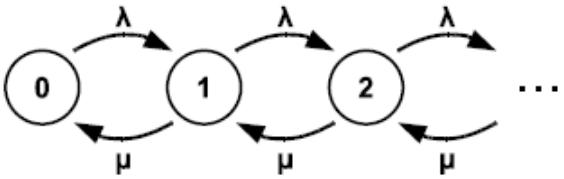

Figure 3.3: Markov process of the M/M/1 queue

Figure 3.3, we can derive the equilibrium distribution for the $M/M/1$ queueing model by using the local balance equations and the normalization condition,

as discussed in Section 3.6, as

$$\pi_i\lambda = \pi_{i+1}\mu \quad \Rightarrow \pi_{i+1} = \frac{\lambda}{\mu}\pi_i = \rho\pi_i$$

$$\sum_{\imath=0}^{\infty} \pi_i = \pi_0 \sum_{i=0}^{\infty} \rho^i = 1 \tag{3.13}$$

$$\Rightarrow \pi_0 = \left(\sum_{i=0}^{\infty} \rho^i\right)^{-1} = \left(\frac{1}{1-\rho}\right)^{-1} = 1-\rho, \quad \text{if } \rho < 1. \tag{3.14}$$

It can be shown that the system is stable when $\rho < 1$. The equilibrium distribution is a geometric distribution,

$$P\{X = i\} = \pi_i \;\; = \;\; (1-\rho)\rho^i, \quad i = 0, 1, 2, \cdots \tag{3.15}$$

$$\tag{3.16}$$

The mean number of jobs in the system, $E[X]$, is computed by

$$E[X] \;\; = \;\; \frac{\rho}{1-\rho}. \tag{3.17}$$

.

Applying Little's formula to (3.17) yields the mean sojourn time of a job in the system,

$$E[T] = \frac{E[X]}{\lambda} = \frac{1}{\mu - \lambda}. \tag{3.18}$$

If the distribution of either inter-arrival or service time is not exponential and the scheduling policy is FIFO, the above results do not hold any more. However, for PS scheduling the formula remains the same although the service time distribution changes. This means that the above formulae also hold for the $M/G/1-PS$ queue, to be discussed in the following section.

3.8 M/G/1-PS queue

In this section, we consider the $M/G/1-PS$ queueing system with Poisson arrival process (M), general service time distributiom (G), a single server $n = 1$, an infinite queue and the processor sharing scheduling discipline.

We assume that new jobs arrive according to a Poisson process with arrival rate λ and the service time distribution can be a general distribution with mean value $1/\mu$. Service times are assumed to be independent and identically distributed (IID) with a continuous distribution,

$$P\{S \leq x\} = F(x) = \int_0^x f(y)\, dy.$$

The processor sharing model is very useful for analysing time-sharing systems and for modeling queueing networks. The processor sharing model, as discussed in Section 3.3, can be interpreted as a queueing system where all jobs are served simultaneously by the server.

The $M/M/1$ queue, discussed in the previous section, is a special case of the $M/G/1$ queue where the service times are IID and exponentially distributed. The state transition diagrams are identical for the classical $M/M/1-FIFO$ system and for the $M/M/1-PS$ system. Thus, the performance measures based on state probabilities are identical for the two systems [12]. In addition, due to the insensitivity property of the PS discipline, discussed in Section 3.3, the results derived for the PS policy in the case of the $M/M/1$ queueing system are valid also for the more general $M/G/1$ system. Thus, the queue length distribution and the mean sojourn time of the $M/G/1-PS$ queue is the same as for the $M/M/1$ queue. .

Chapter 4

Flow level modelling of elastic traffic

For the analysis of a telecommunication system, a model of the system considered must be set up. The characteristics of different kinds of systems can be modeled by using some of the common methods from teletraffic theory. Teletraffic theory is used to model the telecommunication system from the traffic point of view. For data traffic, we can use queuing theory to illustrate and model traffic flows either at packet or flow level. In this chapter, an overview of flow level modelling for elastic traffic is explored based on [13, 14, 15, 16, 17]. In the first section, the concepts related to modelling of data traffic are introduced. Bandwidth sharing of TCP flows is introduced in Section 4.2. Flow level modelling of cellular systems is discussed in Section 4.3.

4.1 Modelling of data traffic

Modelling of data traffic is commonly performed at two different time scales. At the so-called packet level, individual packets are modelled. On the other hand, flow level models characterize end-to-end flows at the transport layer.

4.1.1 Packet level modelling

Data traffic at packet level consists of packets. In Internet, packets corresponds to IP packets that are routed through the network. Individual packets compete with each other for the processing and transmission resources. This technique is called statistical multiplexing. The packet is characterized by its length. The traffic in this case is modelled by considering the following two parameters.

- Packet arrival process specifies at which moments new packets arrive to the system.
- Packet length distribution shows how long the arriving packets are.

We can model the link based on the service rate of the system. The service rate depends on the link capacity and the average packet length.

4.1.2 Flow level modelling

In a longer time scale, data traffic may be thought to consist of flows. A flow is defined as the sequence of packets pertaining to one instance of the same application [14]. When studying the performance measures of elastic traffic, it is preferred to model in terms of flows rather than packets because of two reasons. The first one is the complexity of the packet arrival process, as it has proven to be very difficult to derive a packet level traffic characterization which is useful for performance modelling [16]. The other one is related to the examination of the quality of service experienced by users. Users of elastic applications are generally not sensitive to the end-to-end delay of each packet, but to the time necessary to transfer the entire document, equal to the response time of the associated flow [14].

We have two kinds of flows based on whether the traffic flow rate is dependent on the traffic conditions or not. These are elastic flows and streaming flows. Elastic traffic consists of adaptive TCP flows, such as the transfer of digital documents, where the transfer rate and the duration depend on the current traffic state dynamically. Quality of service for these flows is measured by the rate and duration of flows.

On the other hand, streaming traffic consists of UDP flows. UDP does not control the transmission rate based on the traffic condition as TCP does. In other words, in UDP transmission rate is independent of traffic conditions in the network. Streaming flows are produced by audio and video applications.

The statistical properties of elastic flows are characterized by the flow arrival and flow size distributions. In many cases, flows are generated within sessions. A session may be defined as an alternating series of flows and "think-times". The statistical properties of a session, including flow size distributions and correlations between successive flows and think-times, can be complex and clearly depend on the underlying application. However, these statistical properties are independent from one session to another [14]. This independence leads to a Poisson session arrival process when the number of users is large and no one user generates a significant proportion of the overall traffic. On the contrary, it has been shown in [18] that the flow arrival process tends to be bursty and self-similar in some cases. The main reason for this is that the number of flows per session has a heavy-tailed distribution. However, in certain circumstances it may be appropriate to suppose flows arrive according to a Poisson process. This would be the case, for example, when flows correspond to a large number of independent sessions and the spacing of flows within a session is large compared to the average inter-flow interval.

Measurements of the size of documents such as Web pages and FTP files show that their distribution has a heavy tail [6]. As a result, the large majority of flows are very small while most of the traffic in bytes is contained in large flows. The Pareto distribution, discussed in Section 3.1, can be used to model the flow size distribution of elastic traffic.

4.2 Bandwidth sharing of TCP flows

TCP controls the amount of data that is to be sent to the remote peer on a specific connection by two concurrent mechanisms. The first one is flow control which prevents the source from overloading the destination. The other one is called congestion control. Bandwidth sharing between TCP connections

(flows) is realised by the TCP congestion control algorithm. The congestion control algorithms in TCP are implemented to exploit the available capacity while adjusting the sending rate of competing transfers [17]. Therefore, it is responsible for allocating the bandwidth between different flows.

It is shown in [16] that when several permanent TCP connections use the same bottleneck link, the bandwidth is shared in inverse proportion to RTT and that there is no wasted bandwidth. It has also been proved that the throughput of the flow is quite dependent on the number of flows in the link. Experiments have proved that if all flows have the same RTT, TCP tends to share bandwidth equally among the flows in progress, at least for the larger flows.

In the fair sharing, it has been observed that the number of flows in progress is comparable to the number of customers in an PS queue. Since the behaviour of the flows in fair sharing discipline can resemble the flow behaviour achieved by TCP, the performance of TCP connections can be modelled as a PS queue.

4.3 Modelling of cellular systems

In this section, the wireless system model and the traffic model is introduced. Data services constitute a significant part of today's HSDPA networks. However, the introduction of smart phones has resulted in a drastic increase in data demand. To tackle this, LTE has been introduced, as discussed in Section 2.2. In LTE, eNodeB decides the allocation of physical resource blocks (PRB) to its users. PRB is the smallest unit that can be allocated to a user in one subframe [1]. Similarly, in WCDMA based systems, users within a base station coverage area are allocated with CDMA codes. In such systems, the base station schedules all the transmission of users. In both cases, WCDMA and LTE, the scheduler determines the way radio resources are shared among users.

A single cell with a single downlink channel, whose resources are time-shared between active users, can be viewed as a queueing system at flow level. Each active user is independently and uniformly distributed in the cell at a random

distance R. Flows arrive at base station according to a Poisson process with rate λ and flow sizes are independent and identically distributed with mean $E[X]$.

We assume that the scheduler is not channel aware so that it does not take into account the radio environments such as user mobility, shadowing and multi-path reflections. This assumption results in the data rate to be constant for the duration of the flow. Therefore, the date rate of a user i depends only on the distance r from base station to user i. We denote C_0 and r_0 as the maximum peak rate and the maximum distance at which the peak rate is achieved. The maximum rate, which depends on channel bandwidth and coding efficiency, decreases after r_0 due to path loss. Therefore, mathematically the peak rate function can be expressed as

$$c(r) = \begin{cases} c_0 & \text{if } r < r_0, \\ c_0 \left(\frac{r_0}{r}\right)^{\beta} & \text{otherwise,} \end{cases}$$

where β is the path-loss exponent which characterizes the radio environment.

The rate $c(r)$ represents the transmission rate of the system at a distance r from the base station. The mean service time of a flow at a distance r can be expressed as

$$E[S|R = r] = \frac{E[X]}{c\,(r)}. \tag{4.1}$$

The mean service time of a flow is thus

$$\frac{1}{\mu} = E[S] = E[X]E\left[\frac{1}{c(R)}\right]. \tag{4.2}$$

With the given assumptions, the system corresponds to an $M/G/1$ queue with Poisson arrival of flows at rate λ and a mean service time of $E[S]$. The stability limit of the system is given by $\rho = \lambda E[S] < 1$. When the round-robin scheduling policy is used, the system can be modelled as an $M/G/1 - PS$ queue. For such a case the explicit expression for the throughput of a user at

a distance r is derived in [15] as

$$\gamma(r) = c\,(r)\,(1-\rho)\,. \tag{4.3}$$

This implies that the flow throughput decreases linearly in the cell load. In addition, the mean delay conditioned on the distance of a user located at r from the base station is shown to be

$$T(r) = \frac{E[X]}{c\,(r)\,(1-\rho)}. \tag{4.4}$$

The simplified assumption, not taking into account radio parameters, enables us to have a queueing model for cellular network in an idealized situation. However, in the real world there are many factors that affect the bandwidth sharing. The capacity of cellular network may vary greatly because of the fluctuation of the transmit power and the increase of interference. Besides, different RTTs for individual flows may make such balanced fairness even impossible to realize in the cellular network. However, the ideal and simple $M/G/1-PS$ queue still can model the wireless network to some extent and it provides some explicit and powerful formulae to describe the flow level performance.

Chapter 5

Distributed server system

In this chapter, we consider some of the important existing task assignment policies for distributed systems based on [9, 19, 8, 20, 21, 22, 23, 24, 25]. In the first section, a distributed server system is introduced. In Section 5.2, we examine the classical static policies, such as Random and Round Robin, and size-based policies such as SITA (Size Interval Task Assignment) variants and TAGS (Task Assignment based on Guessing Size). Section 5.3 considers state dependent policies. At the end, in Section 5.4, we report known results of the task assignment policies.

5.1 Introduction

A distributed server system, dispatching system, consists of a dispatcher unit and a collection of parallel servers. The dispatcher applies a dispatching policy called task assignment policy, which acts as a rule for assigning jobs to the parallel servers. A job can be a request for a file on a web page, or a complex computation to be performed. The policy decides when and to which server an incoming request should be routed. The dispatcher then immediately routes the request to one of the servers for processing using the specific task assignment policy. The choice of task assignment policy has a significant effect on the system performance. Thus, designing a distributed server system often boils down to choosing the "best" task assignment policy for the given model

and user requirements [9].

There are several metrics, performance goals, which can be used as an objective to choose a specific task assignment policy. One of the basic metrics is the mean sojourn time the job spends in the system. Another performance goal is to reduce the variance in the mean delay. This is because the more the variance is reduced the more the mean delay becomes predictable. A third performance goal is fairness where all kinds of jobs, whether short or long, should experience the same expected slowdown.

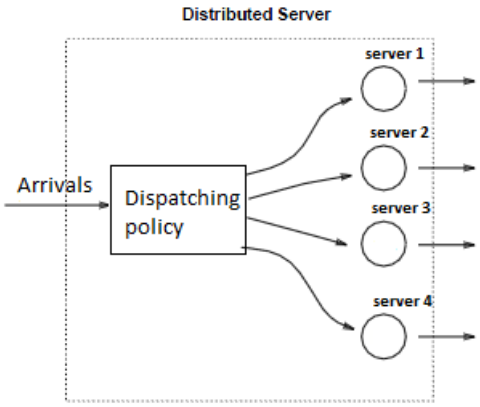

Figure 5.1: Model for distributed server with dispatching policy

Figure 4.1 shows an example of a distributed server system consisting of a dispatcher unit and four servers. Arriving jobs are dispatched to the specific servers. With the given model of a distributed server system, we have the so-called dispatching problem.

The dispatching problem is a joint-optimization problem of two interacting policies: (i) A dispatching policy assigning a queue for each job immediately upon arrival, and (ii) an internal scheduling policy of the queues, i.e., queuing discipline or service order [19]. It is assumed that the job sizes are independently and identically distributed according to a general job size distribution and that the arrival process of jobs to the servers is a Poisson process.

There are several task assignment policies proposed for distributed systems. These policies can be classified into two categories based on the information that the dispatcher uses at the time of decision.

- *Static policies* In these policies the dispatcher's decision does not depend on the states of the queues or past decisions.
- *Dynamic policies* These policies are dynamic policies where the assignment decision uses knowledge of the state of the servers.

5.2 Static policies

In this section, we consider two static policies, Random and Round-Robin, which are state independent. In both policies, the load is equally balanced among the servers. In addition, we also consider two size-based policies.

5.2.1 Random

Random assignment is a static policy, known as Bernoulli splitting, where an incoming task is sent to server i with probability $1/h$, where h is the number of servers in the system. This means each task is assigned to each host with equal probability. As a result, all queues have the same arrival rate. This policy equalizes the expected number of jobs at each server if the servers are identical. However, when the servers are not identical, the optimal splitting probabilities become non-uniform.

5.2.2 Round-Robin

In the Round-Robin assignment the dispatcher rotates between the queues in some predefined order. For identical servers jobs are assigned to servers in a cyclical fashion with the ith job being assigned to server i mod h. This policy also equalizes the expected number of jobs at each server, and has slightly less variability in inter-arrival times than the Random policy [8]. Both Random and Round-Robin policies are frequently used as a base line to compare with other task assignment policies.

5.2.3 Size-based policies

In size-based assignment, all tasks within a certain size range are sent to an individual server [20]. This means the workload is partitioned into distinct size ranges with each size range associated to a specific server. This policiy assigns jobs in a way that "short" jobs are dispatched to one server, "medium" jobs to another server and "long" jobs to the third server. Although being a static policy, size-based policies perform well under very high task size variation for FIFO scheduling. The advantage of using the Size-based policy, in the case of a highly variable job size distribution, is that it isolates short jobs from long ones. This prevents short jobs from getting stuck behind long jobs, thereby greatly reducing the mean response time when the scheduling displine is FIFO.

These size-based policies can be further classified based on what knowledge about the arriving task size is known at the dispatcher. Some policies assumes that a task's size is known a priori at the dispatcher, and as such can assign the task directly to the server that is responsible for servicing tasks in that range. This obviously restricts the application of these policies to domains where exact (or reasonably accurate) a priori knowledge of a task's size is available. Other size-based policies have less restrictive assumptions regarding what information is available at the dispatcher. Policies such as TAGS [21] assume no knowledge of a task's size at the dispatcher.

Size interval task assignment (SITA)

SITA-E (Size Interval Task Assignment with Equal Load) is a size-based approach proposed by Harchol-Balter et al. [8]. SITA-E associates a unique size range with each server in the distributed system and a task is sent to the appropriate server based on its size. These size ranges are chosen specifically to equalise the expected load received at each server. The cutoff points are computed once, given the distribution, and then the dispatcher unit needs only keep a record of these cutoff points for implementing the SITA-E policy [8].

There are also other variants of SITA policies. SITA-V (Size Interval Task Assignment with Variable Load) purposely operates the server hosts at dif-

ferent loads. SITA-V directs smaller tasks to lighter-loaded servers. SITA-V, like SITA-E, assigns tasks to a given host based on their size. However, SITA-V exploits the heavy-tailed property of the task size distribution by running the vast majority of tasks (i.e. the small tasks) on lightly-loaded hosts, while running the minority of tasks (the larger sized tasks) on the heavily-loaded hosts. The result is that SITA-V provably decreases the mean task slowdown by significant factors.

Task assignment based on guess (TAGS)

The SITA based approaches considered above assume that the exact service requirement is known at the dispatcher before the dispatchig is done. However, in many practical cases the task service requirement is not known until execution time on a given server. Task Assignment based on Guessing Size (TAGS) [21] assumes no prior knowledge of a tasks service requirement. The TAGS approach works by associating a processing time limit with each server. Tasks are executed on a host up until the designated time limit associated with that server. If the task has not completed by this point, it is killed and restarted from scratch at the next host. These cutoffs are a function of the distribution of task sizes and the outside arrival rate, and can be computed to optimise certain metrics, such as the mean sojourn time. The design of the TAGS policy purposely exploits properties of the heavy-tailed distribution, such as decreasing failure rate where the longer a task has run, the longer it is expected to run and the fact that a tiny fraction (less than 1 %) of the very longest tasks can make up over half the load.

5.3 State dependent policies

In state dependant policies, the dispatching decisions depend on the state of the system. In this section, we consider two kinds of state dependant policies. First, in Section 5.3.1, we introduce policies where the dispatching decisions depend only on the state of the queues in the system. Such policies do not take into account the size of the arriving jobs. In Section 5.3.2, policies that

take into account both the state of queues in the system and the size of the arriving jobs are introduced.

5.3.1 Policies independent of the arriving job

The policies that dispatch the arriving jobs to the server with the least amount of work remaining are independent of the arriving job. These policies include join the shortest queue (JSQ), least work load (LWL) and Central Queue.

Join the shortest queue (JSQ)

JSQ is the prime example of a state-dependent policy, where the dispatcher chooses the queue with the least number of jobs [24]. This policy tries to equalize the instantaneous number of jobs at each server. JSQ is the optimal policy for exponentially distributed job sizes and identical servers [24].

Least work load (LWL)

LWL assigns each incoming job to the server with the least total work, where the work at a given server is the sum of the remaining service times of jobs queued at that server. LWL assumes that the service requirement of the jobs is known a priori. For a distributed system with the FIFO scheduling discipline, it minimizes the waiting time for the new job.

Central Queue

The Central-Queue policy holds all the incoming jobs in a FIFO queue at the dispatcher until some of the available servers is free to accept the arriving job. It is the same as the basic $M/G/n$ queue. Such a policy has proved to be equivalent with the LWL dispatching policy combined with the FIFO scheduling policy, showing that equivalent performance can be obtained without any prior knowledge of a task's size [21]. There needs to be constant feedback between the dispatcher and the servers. This is because each server must notify

the dispatcher that there is no job when ever that server is empty.

5.3.2 Policies dependent of the arriving job

The dispatching decisions in policies dependent of the arriving job depend on both the state of queues in the system and the size of the arriving job. The policies considered below are minimum expected delay (MED), Least flow-time first (LFF) and Myopic (MP).

Minimum expected delay (MED)

The MED routing policy, introduced in [25], is defined as follows. Let $N_i(t)$ be the number of jobs at server i at time t, and define $u_i(t) = (1 + N_i(t))/\mu_i$, where μ_i is the mean service rate of server i. Note that $u_i(t)$ corresponds to the expected delay for a job that enters queue i at time t if service times are exponential and the server applies the FIFO discipline. Let us define $u^*(t) = \min\{u_i(t), i = 1, ..., K\}$, where K is the number of servers in the system. Upon arrival of a job at time t, the job joins a server j where $u_j(t) = u^*(t)$. If a tie occurs, the job chooses the server with the highest service rate. When all service rates are equal, this MED routing policy reduces to the classic JSQ policy.

Least flow-time first (LFF)

The task assignment in LWL, discussed above, is only based on absolute server loads and it assigns a job to the least loaded server. However, in a heterogeneous server farm, servers may have different processing capacities. LFF, proposed in [23], dynamically assigns a job based on the remaining processing time of jobs on servers and the sojourn time of the new job if it were executed on each server. The sojourn time of a new job can be estimated based on the server loads and the processing capacities of servers before it is assigned. It assigns the task to the server with the minimum sojourn time when the FIFO scheduling discipline is applied.

Myopic (MP)

The myopic policy was introduced in [22] for the case of a system of processor sharing servers with Poisson arrivals and general service times when the dispatcher knows the remaining service requirement for all jobs present in the system and the service requirement of the incoming job. When the job sizes are known, it is possible to compute the additional cost in terms of cumulative sojourn time for each possible action, and consequently to choose the optimal queue. It originates from the assumption that no further jobs arrive after the job that has just arrived.

5.4 Known results of dispatching policies

Harchol-Balter et al. [8] have shown that the performance of the system under the Random and Round Robin policies is similar for FIFO scheduling and that both policies perform much more poorly than the others (SITA-E and dynamic policies) for the case when the task sizes are highly variable. Harchol-Balter has found that when the task sizes are not highly variable, the dynamic policy (Least Work Load) is preferable. However, when task sizes show higher degree of variability, SITA-E performs better than both the classical static and dynamic policies. It can be concluded that the best choice of task assignment policy depends critically on the task size distribution, and in particular on the variability of this distribution. In addition, Feng and Misra [26] found that when the dispatching system is comprised of a set of homogeneous servers, then SITA is optimal amongst all static policies.

Harchol-Balter [21] proved that TAGS policy improves the performance of LWL and static policies for the case when the job size distribution is Bounded Pareto and the service discipline is FIFO. Variance reduction and load unbalancing were the reasons, mentioned in [21], to justify why TAGS performs better than LWL. Although size-based approaches have shown encouraging results, both SITA and TAGS have common problems. One problem with size-based approaches is that they may yield an unbalanced load for the servers. This is because as the task assignment policies are dictated by size, they do

not consider the load of the server they are assigning to. Therefore they may assign a large task to an already overloaded server, while leaving other servers under-utilised. On the other hand, dynamic policies improve resource utilisation by balancing the load, and hence reduce the probability of a server being idle.

Bonomi [22] has shown that JSQ is not optimal for non-exponential service time distributions for the case when the service discipline is processor sharing. In addition, Crovella and Harchol-Balter [8] showed that, for distributed systems with heavy-tailed workloads employing a processor-sharing policy at the hosts, SITA-V reduces the mean slowdown to levels far below those that a balanced-load policy would achieve.

Chapter 6

Load balancing in heterogeneous networks

In this chapter, we consider the problem of load balancing in heterogeneous networks in detail by applying the techniques discussed in the previous chapters. In Section 6.1, we introduce the research problem, to balance the load in heterogeneous networks, and present the traffic allocation model for the system. In Section 6.2, we discuss a probabilistic allocation method, and in Section 6.3 we introduce the optimal probabilistic approach to the load balancing problem. Finally, in Section 6.4, we present dynamic policies that depend on the state of the system under consideration.

6.1 System model

The heterogeneous networks, as illustrated in Figure 2.5, consist of a macrocell that is supported by different types of microcells which are deployed to improve spectral efficiency per unit area. These different cells have their own characteristics with respect to handling data traffic. From the traffic point of view, we can consider each cell, whether it is macrocell or microcell, as a server which has its own service rate. Much of the traffic is elastic, so each cell can be modelled as a $M/G/1-PS$ queuing system, discussed in Section

3.8, by assuming the arrival process to be a Poisson process and the incoming flow sizes obey a general distribution.

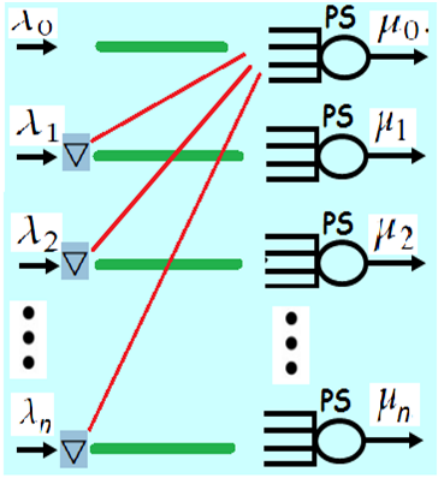

Figure 6.1: Load balancing model in heterogeneous networks

The diagram in Figure 6.1 shows how the traffic allocation system is modelled in heterogeneous networks. The model consists of a single macrocell and n microcells, where each cell has its own dedicated wireline connection for the traffic. Thus, each cell can be assumed to act as a server which has its own arrival and service rate. It is assumed that the macrocell has a dedicated arrival rate, λ_0, and a service rate of μ_0. Similarly, for the n microcells we have n different arrival rates λ_i and n different service rates μ_i. It is also assumed that the service rates of microcells are larger than that of a macrocell,

$$\mu_i \geq \mu_0 \text{ for all } i. \tag{6.1}$$

In an *unbalanced system*, each incoming flow of a user within a specific cell is served by that cell independent of whether the cell is congested or not. However, in the *balanced system* flows arriving in any microcell i may be served either by the local microcell i or macrocell. In this chapter, we introduce different static and dynamic load balancing mechanisms.

There exists a stable load balancing policy if and only if the queueing system,

presented in Figure 6.1, is stable. The necessary condition for this stability is

$$\lambda_0 + \sum_{i=1}^{n} (\lambda_i - \mu_i)^+ < \mu_0. \tag{6.2}$$

where $(x)^+$ is equal to x, if $x > 0$; otherwise it is equal to 0. Note that the sum represents the excess traffic from n microcells that the macrocell must handle.

6.2 Probabilistic allocation

In this section, we consider a static probabilistic allocation policy similar to what is described in Section 5.2.1. For heterogeneous networks that consists of a single macrocell and n microcells, a probabilistic allocation policy is described by the probability vector $\mathbf{p} = (p_1, p_2, ..., p_n)$. In this policy, the incoming flow is assigned to the microcell i with probability p_i and to the macrocell with probability $1 - p_i$. Figure 6.2 shows the static traffic allocation in heterogeneous networks based on a probabilistic approach.

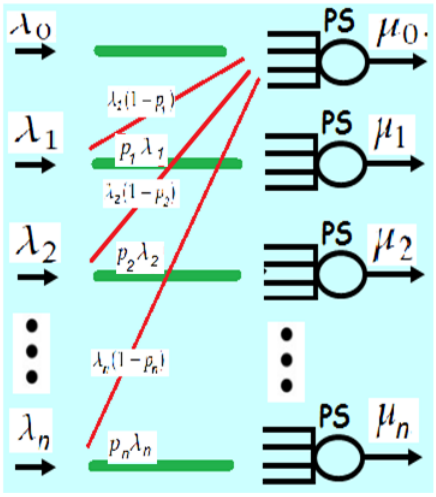

Figure 6.2: Static traffic allocation in heterogeneous networks

From queueing theory we know that, for any work-conserving scheduling discipline, server i is stable if the load, $\rho_i < 1$. Due to the splitting property of the Poisson process, discussed in Section 3.5, the arrival process to each of the queues will also be Poisson. Then, we can say that the system model, given in

Figure 6.2, is composed of $n+1$ parallel M/G/1 queues with a single macro cell and n microcells. Therefore, we have the following stability condition

$$\lambda_i p_i < \mu_i \text{ and } \lambda_0 + \sum_{i=1}^{n} \lambda_i (1 - p_i) < \mu_0 \text{ for all } i. \tag{6.3}$$

The first part of (6.3) is the stability limit for each microcell i and the second part is for the macrocell. Thus, (6.3) is the necessary and sufficient condition for the stability of the system under consideration.

For a stable system, we have the expression for the mean delay of the model as

$$E[T] = \frac{\lambda_0 + \sum_{i=1}^{n} \lambda_i (1 - p_i)}{(\lambda_0 + \sum_{i=1}^{n} \lambda_i)(\mu_0 - (\lambda_0 + \sum_{i=1}^{n} \lambda_i (1 - p_i)))} + \sum_{i=1}^{n} \frac{p_i \lambda_i}{(\lambda_0 + \sum_{i=1}^{n} \lambda_i)(\mu_i - p_i \lambda_i)}. \tag{6.4}$$

6.3 Optimal probabilistic allocation

The question at hand is to find an optimal probability vector $\mathbf{p}^* = (p_1^*, p_2^*, ..., p_n^*)$ so that (6.4) becomes minimized. These optimal probabilities depend on the service rate of the macro and microcells, the dedicated arrival rate of macrocell and the arrival rates of microcells. The problem becomes a mathematical optimization problem with a nonlinear objective function, $E[T]$. Since the objective function is nonlinear, we have a nonlinear programming problem.

In this section, we consider the problem of finding the optimal probabilities in two traffic scenarios. In the first case, we consider the *symmetric traffic scenario* where the arrival and service rate of the different micro cells are identical. For this case the optimal allocation probability can be solved analytically. The second one is the *asymmetric traffic scenario* where the traffic parameters are different for different micro cells. Mathematical optimization and numerical methods are used to solve the optimal probability value for this case.

6.3.1 Symmetric case

The expression for the mean delay, $E[T]$, that is mentioned in (6.4) is for the general case where the parameters such as arrival rate, service rate and optimal allocation probabilities are different for different microcells. When considering the symmetric case, we assume that the parameters that are related to the microcells are equal, i.e, $\lambda_1 = \cdots = \lambda_n = \lambda$ and $\mu_1 = \cdots = \mu_n = \mu$. Due to this symmetry, we may restrict the study to *symmetric policies* for which $p_1 = \cdots = p_n = p$. The stability condition for the symmetric policies is given as

$$\lambda p < \mu \text{ and } \lambda_0 + n\lambda\left(1-p\right) < \mu_0 \ . \tag{6.5}$$

This assumption simplifies the expression for the mean delay to

$$E[T] = \frac{\lambda_0 + n\lambda\left(1-p\right)}{\left(\lambda_0 + n\lambda\right)\left(\mu_0 - \left(\lambda_0 + n\lambda\left(1-p\right)\right)\right)} + \frac{p\lambda n}{\left(\lambda_0 + n\lambda\right)\left(\mu - p\lambda\right)}. \tag{6.6}$$

To find an optimal value for p^* which minimizes the mean delay, $E[T]$, we will first check the convexity of the function by looking into whether $\frac{\partial^2 E[T]}{\partial^2 p} \geq 0$. The second derivative for $E[T]$ is given as

$$\begin{aligned}\frac{\partial^2 E[T]}{\partial^2 p} = &\frac{2n^2\lambda^2}{\left(\lambda_0 + n\lambda\right)\left(\mu_0 - \left(\lambda_0 + n\lambda\left(1-p\right)\right)\right)^2} + \frac{2np\lambda^3}{\left(\lambda_0 + n\lambda\right)\left(\mu - p\lambda\right)^3} \\ &+ \frac{\left(2n^2\lambda^2\right)\left(\lambda_0 + n\lambda\left(1-p\right)\right)}{\left(\lambda_0 + n\lambda\right)\left(\mu_0 - \left(\lambda_0 + n\lambda\left(1-p\right)\right)\right)^3} + \frac{2n\lambda^2}{\left(\lambda_0 + n\lambda\right)\left(\mu - p\lambda\right)^2}.\end{aligned} \tag{6.7}$$

The above expression, (6.7), is positive in the whole domain of p. Therefore, we have a convex function where the first order condition, $\partial E[T]/\partial p$, is sufficient for a minimizer of a convex function. So, by using the first derivative approach to (6.6), we can determine the allocation probability value for p as

$$p = \frac{\sqrt{\mu_0}\mu + \sqrt{\mu}\left(\left(\lambda_0 + n\lambda\right) - \mu_0\right)}{\sqrt{\mu_0}\lambda + \sqrt{\mu}n\lambda}. \tag{6.8}$$

From (6.8), it can be observed that $p \geq 0$ since, due to (6.1), we have

$$p > \frac{\sqrt{\mu_0}\mu - \sqrt{\mu}\mu_0}{\sqrt{\mu_0}\lambda + \sqrt{\mu}n\lambda} = \frac{\sqrt{\mu_0}\sqrt{\mu}\left(\sqrt{\mu} - \sqrt{\mu_0}\right)}{\sqrt{\mu_0}\lambda + \sqrt{\mu}n\lambda} \geq 0. \tag{6.9}$$

However, we have a constraint that the value of p should not be greater than 1, $p \leq 1$. Thus, by using the convexity property we have

$$p^* = \min\left(p, 1\right), \tag{6.10}$$

where p is given in (6.8).

6.3.2 Asymmetric case

In the previous section, we considered the symmetric case by assuming that the parameters related to the microcells are equal. However, in heterogeneous networks we have different kinds of microcells such as femtocells, picocells and relays where the arrival rates and the capacities of these cells could be different. In this section, we consider the asymmetric case where the parameters such as arrival rates, service rates and optimal probabilities are different for different microcells. The necessary and sufficient stability condition for this case is the same as what have been is give in (6.3).

For the asymmetric case, the expression for the mean sojourn time will be the same as in (6.4). The objective is to determine an optimal allocation strategy so that the mean sojourn time is minimized. This can be stated as

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \; E[T]. \tag{6.11}$$

We can rewrite (6.11) as a global optimization problem, in which the dispatcher in the microcell decides where each arriving flow to the microcell gets service so as to minimize the mean sojourn time the flow spends in the system. The

optimization problem can be formulated in a standard form as

$$\begin{aligned} \text{minimize} \quad & E[T] \\ \text{subject to} \quad & \lambda_i p_i - \mu_i \leq 0, \qquad \text{for all } i \in \{1, ..., n\} \\ & \left(\lambda_0 + \sum_{i=1}^{n} \lambda_i (1 - p_i)\right) - \mu_0 \leq 0, \\ & p_i - 1 \leq 0, \qquad \text{for all } i \in \{1, ..., n\} \\ & p_i \geq 0, \qquad \text{for all } i \in \{1, ..., n\}. \end{aligned} \tag{6.12}$$

Since the objective function, $E[T]$, and the constraints are convex, the optimization problem of (6.12) can be stated as a standard convex optimization problem. Treating the problem as a convex optimization is beneficial due to the following crucial properties of convex optimization problems:

1. *No local minima*: any local optimum is necessarily a global optimum;
2. There exists efficient and reliable numerical methods that can handle very large and practical engineering problems.

The convex optimization problem at hand is an inequality constrained minimization problems. There are several algorithms that can be used to solve such kind of problems. Interior point algorithm is used in this thesis to find the optimal allocation strategy as implemented in Mathematica. The goal of interior point algorithm is to formulate the inequality constrained problem, such as (6.12), as an equality constrained problem.

6.4 Dynamic policies

Dynamic policies are, as discussed in Section 5.3, state dependent policies. The state of the system, $\mathbf{s}$, depends on the chosen policy. It may be the number of flows in the system or the remaining workload of the flows.

In this section, we consider four dynamic polices called JSQ, MJSQ , LWL and

MP for the load balancing model shown in Figure 6.1. When a flow arrives to a specific microcell i, the dispatcher allocates it to the specific queue based on the chosen policy. We define $\Delta_i(\mathbf{s})$ as the action that the dispatcher, that employs a given policy, takes when the system is in state $\mathbf{s}$.

6.4.1 Join the shortest queue

In the join the shortest queue policy (JSQ), the task assignment is determined by comparing the number of flows. The state of the system can be determined by the number of flows in the macrocell and microcell in which the incoming flow arrives. So the system state is $\mathbf{s} = (n_0, n_i)$, where n_i is the number of flows being served in server i. The JSQ policy can be stated as

$$\Delta_i(\mathbf{s}) = \text{argmin}\{n_0, n_i\}. \tag{6.13}$$

When the number of flows in both the macrocell and the microcell i are equal, the policy is implemented to assign the arriving flow to the microcell i.

6.4.2 Modified join the shortest queue

The modified join the shortest policy (MJSQ) is similar to the MED policy, which was discussed in Section 5.3.2. However, the MJSQ doesn't take into account the size of the arriving flow. It is a modified version of the basic JSQ policy where the number of flows in the server is scaled with the service rate of that server. Therefore, additional information, the service rate of the macrocell and microcells, is required to determine the state of the system. So the system state becomes $\mathbf{s} = ((n_0, \mu_o), (n_i, \mu_i))$, where μ_i is the service completion rate of server i. We state MJSQ as

$$\Delta_i(\mathbf{s}) = \text{argmin}\{n_0/\mu_0, n_i/\mu_i\}. \tag{6.14}$$

6.4.3 Least work load

The least work load (LWL) policy assigns the arriving flows to the microcells based on the comparison of the remaining workload in time units in the micro and macrocell. This policy assumes that the unfinished work, u_i, of the flows in both macrocell and microcell is known at the arrival of the new flow. The system state is expressed as $\mathbf{s} = (u_0, u_i)$. We state the LWL policy as

$$\begin{aligned} \Delta_i(\mathbf{s}) &= \operatorname{argmin}\{u_0, u_i\}, \\ \text{with } u_i &= \sum_{j=1}^{n_i} r_{ij}, \end{aligned} \tag{6.15}$$

where r_{ij} is the remaining service requirement of the jth flow at server i and n_i is the number of flows that are being served in that server.

6.4.4 Myopic policy

As discussed in Section 5.3, the myopic policy (MP) allocates the arriving flow based on the additional cost introduced by the flow. It compares, between the macro and microcell, the additional delay in the system experienced by the incoming flow. The policy assumes that the remaining service requirements of the flows and the service requirement of the incoming flow are known.

Let the state of the system for a specific PS server just before the arrival of a new flow with service requirement r^* be given by

$$\mathbf{s} = \{n, r_1, r_2, \cdots, r_n\}, \quad r_1 \leq r_2 \leq \cdots \leq r_n. \tag{6.16}$$

where n is the number of flows being served in the server. Bonomi [22] showed the explicit formula that can be used to calculate the effect, σ, of the assignment of one additional flow with the assumption that no arrival happens

afterwards. This can be expressed as

$$\sigma = 2\sum_{i=1}^{k} r_i + r^* \left(2n - 2k + 1\right), \tag{6.17}$$

where $k = \max\{i : r_i \leq r^*\}$.

Let the additional cost of allocating the incoming arrival to the macrocell and the ith microcell be σ_0 and σ_i, respectively. Upon the arrival of a flow to the microcell, the dispatcher allocates this flow to the specific cell according to MP as

$$\Delta_i \left(\mathbf{s}\right) \quad = \quad \text{argmin } \{\sigma_0, \sigma_i\}. \tag{6.18}$$

Chapter 7

Numerical results

In this chapter, we examine the performance of the load balancing policies for heterogeneous networks. The performance analysis is based on the simulation method. The system model considered here is the same as described in Figure 6.1. In the first section, we will introduce a limited set of the traffic and system scenarios that help to illustrate how the policies and the system behave in different scenarios. Section 7.2 is about the implementation of the simulator. The performance of the policies under symmetric and asymmetric traffic scenarios will be presented in Section 7.3 and 7.4, respectively. Section 7.5 considers the effect of flow size variation on the implemented policies. In addition, in Section 7.6 we will study the effect of the number of microcells on the performance of load balancing policies. At the end, a short summary about the simulation results is given.

7.1 Traffic and system scenarios

The system under consideration, as discussed in Section 6.1, consists of a single macrocell and a number of microcells. In this chapter, we consider five different traffic and system scenarios. Some other scenarios and the corresponding numerical results are discussed in Appendix A.

The basic system model consisting of two microcells ($n = 2$) is illustrated in

Figure 7.1. We consider four different traffic scenarios as discussed below.

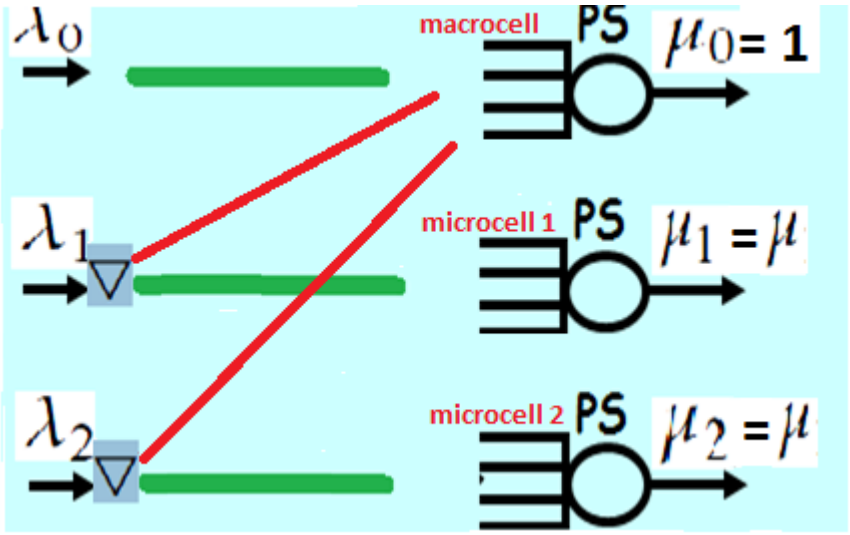

Figure 7.1: Illustration of traffic and system scenario: $n = 2$

In the first two traffic scenarios, we consider symmetric cases where the arrival rate to and the service rates of the microcells in the system under consideration are equal. In the first case, it is assumed that there will be no arrivals in the macrocell, $\lambda_0 = 0$, and the arrival rates to the microcells will be the same, $\lambda_1 = \lambda_2 = \lambda$, where λ is varied. In the second case, we consider the traffic scenario where there will be a constant and identical arrival rate to all the microcells, $\lambda_1 = \lambda_2 =$ constant, and an independent arrival rate to the macrocell, λ_0, which is varied. This traffic scenario is also studied in detail in Section 7.5 to investigate the effect of flow size variation on the implemented policies.

In the other two traffic scenarios, we will consider asymmetric cases where the arrival rates to the microcells in the system under consideration are different. In the third case, like in the first case, it is assumed that there will be no arrivals in the macrocell, $\lambda_0 = 0$, and arrival rate to microcell 1, λ_1, is constant. However, the arrival rate to microcell 2, λ_2, is varied. In the fourth case, there will be a constant but different arrival rate to the microcells, $\lambda_1 = 1, \lambda_2 = 2$, and an independent arrival rate to the macrocell, $\lambda_0 = \lambda$. Table 7.1 summarizes the above four traffic scenarios.

Traffic scenarios	fixed parameters	independent parameters
Traffic scenario 1	$\lambda_0 = 0$	$\lambda_1 = \lambda_2 = \lambda$
Traffic scenario 2	$\lambda_1 = \lambda_2 = 2$	$\lambda_0 = \lambda$
Traffic scenario 3	$\lambda_0 = 0, \lambda_1 = 2$	$\lambda_2 = \lambda$
Traffic scenario 4	$\lambda_1 = 1, \lambda_2 = 2$	$\lambda_0 = \lambda$

Table 7.1: Summary of traffic scenarios

In addition to the above traffic scenarios, in Section 7.6 we look into the effect of increasing the number of microcells on the performance of the load balancing policies.

In the following sections, we will go through the results obtained from studying each of the traffic and system scenarios mentioned above. In all traffic scenarios, we assume that flows arrive to each server i as a Poisson stream with rate λ_i and the flow sizes follow a general distribution. Numerical results consider exponential and bounded Pareto distributions, discussed in Section 3.1, to model the incoming flow size distribution.

7.2 Implementation of the simulator

To study the performance of the dynamic load balancing policies discussed in Section 6.4, a simulator has been developed using Wolfram Mathematica software (version 8.0). The simulator is event based where we consider flow arrivals and departures as events. The length of a single simulation run, which is the stopping condition of the simulator, has been 500 000 arrivals for each traffic scenario. The simulation was repeated 10 times with the same parameters to determine the confidence intervals. For all traffic and system scenarios considered in this chapter, we model flow sizes to follow both exponential and bounded Pareto distributions. For the exponential distribution, we set the mean flow size to 1. For the bounded Pareto case, we keep the mean flow size at 1 and the maximum flow size is fixed at 1000. Table 7.2 presents the parameters used in the simulations.

We simulated the system under each policy by considering the mean number of

Parameter	values
Length of the simulation(time unit)	500,000
Mean flow Size, $E[X]$	1
the largest possible flow size (p)	1000
The shortest possible flow size (k)	chosen so that the mean flow size =1
The shape parameter (α)	2

Table 7.2: Parameters used in simulating the load balancing policies

flows in the system as a performance metric. For the optimal static policy, the mean number of flows in the system can be determined analytically. Therefore, both analytical and simulation results are shown for the optimal static policy specifically. The ratio of the mean number of flows in the system of the dynamic policies to that of the optimal static policy is plotted for each traffic scenario.

7.3 Symmetric scenarios

In this section, the so-called symmetric traffic scenarios will be examined. We will go through results obtained from two symmetric traffic scenarios discussed in Section 7.1. Figures 7.2.a and 7.2.b illustrate Traffic scenario 1 and Traffic scenario 2, to be discussed next, respectively.

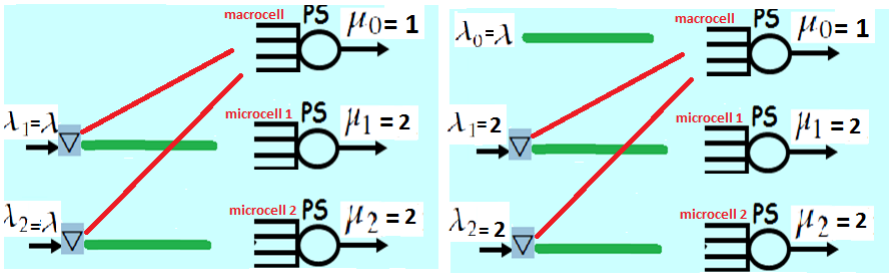

Figure 7.2: Symmetric traffic scenarios: (a) Traffic scenario 1 (left) and (b) Traffic scenario 2 (right)

7.3.1 Traffic scenario 1

In this section, we consider the case in which the number of microcells, n, is only two. In this traffic scenario, we assume that $\lambda_0 = 0$, $\lambda_1 = \lambda_2 = \lambda$. So λ is a free parameter that can be varied up to the stablility limit which will be discussed next. The capacity of the servers, i.e, the service rates, are specified as $\mu_0 = 1, \mu_1 = \mu_2 = 2$.

Stability condition and analytical results

We need to consider the stability condition to find the threshold for the λ value. For the system to be stable, the total arrival rate should be less than that of the service rate. So in this specific case, using (6.2), we have the relation $2\lambda < \mu_0 + 2\mu$. Since $\mu = 2$, we obtain threshold $\lambda^{\max} = 2.5$. However, in an unbalanced system the mean delay goes to inifnity, $E[T] \to \infty$, if $\lambda \geq 2.0$.

Figure 7.3.a shows how the optimal allocation probability (p^*), given in (6.10), varies by changing the value of λ from 0 to 2.5. It can be observed that until the arrival rate, λ, reaches 0.6, the optimal allocation probability is 1. This means that packets are never forwarded to the macrocell when the arrival rate to the microcell is less than 0.6. But when λ starts to increase from 0.6 onwards, it is seen that the allocation probability decreases. Thus, it is advisable to allocate the arriving flows to the macrocell with a probability of $1 - p^*$. Figure 7.3.b also shows the mean sojourn time of a flow in the system for optimal probabilistic allocation policy.

Simulation results and performance comparison

The numerical results concerning the performance of dynamic policies in Traffic scenario 1, obtained by means of simulation, are illustrated in Figure 7.4 for two kinds of flow size distributions: exponential and bounded Pareto (from left to right). The y-axis represents the ratio in the mean number of flows in the system between the implemented policy and the base line optimal static policy, and the x-axis shows the arrival rate to the microcells. The curves in Figure 7.4 from top to bottom correspond to (1) simulated optimal static policy, (2)

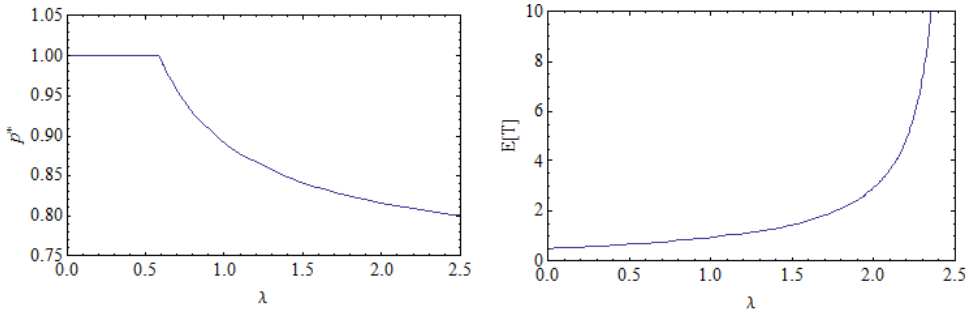

Figure 7.3: Analytical results related to the optimal static policy for Traffic scenario 1: (a) Optimal allocation probability (left) and (b) the mean sojourn time the flow spends in the system (right).

JSQ, (3) LWL, (4) MJSQ and (5) the MP policy. It can be observed that LWL is not insensitive to the flow size distribution. However, other policies tend to have an insensitivity property. There is also an interesting similarity in the performance gain between MJSQ and MP, although MP needs additional information compared to MJSQ. In addition, these two policies have a better performance than other policies.

It can also be observed that the percentage decrease in the mean number of flows (the gain of the load balancing policies) depends on the arrival rates to the microcells, λ. For small values of λ, there is a slight load balancing gain, around 10-15 %, in all policies. However, at higher arrival rates to the microcells, the gain of the load balancing policies is quite large, approximately 50 %.

7.3.2 Traffic scenario 2

In this traffic scenario, similar to Traffic scenario 1, we consider the case where the number of microcells is only two. However, unlike in Traffic scenario 1, we assume that there are dedicated arrivals to the macrocell. In addition, we assume that we have a fixed arrival rate for the microcells. So the parameter $\lambda_0 = \lambda$ is taken as an independent parameter and $\lambda_1 = \lambda_2 = 2$. The service rates of the servers are the same as in Traffic scenario 1.

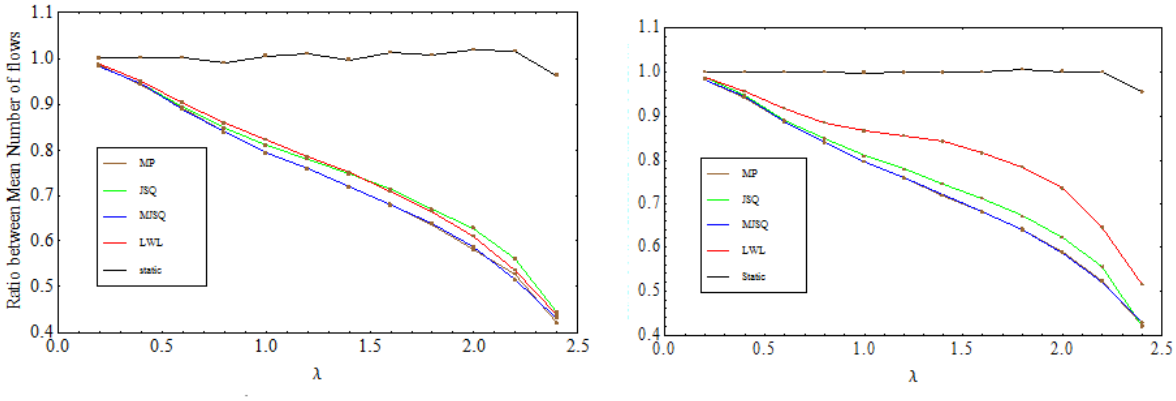

Figure 7.4: Ratio of the mean number of flows in the system between the dynamic and base line optimal static policies for Traffic scenario 1: (a) exponentially distributed flows (left) and (b) bounded Pareto distributed flows (right).

Stability condition and analytical results

The stability condition for this traffic scenario is determined solely by the arrival rate of the macrocell. Thus, the threshold for the stability is $\lambda_0^{\max} = 1$.

Figure 7.5.a shows the optimal allocation probability when the arriving traffic rate in the macro-cell, λ_0, varies from 0 to 1. It can be concluded that increasing the dedicated arrival rate λ_0 results in an increase in the optimal allocation probability which in turn reduces the probability of switching the arrival from micro cells to the macro cell. The mean sojourn time of a flow in the system for the optimal probabilistic allocation policy is shown in Figure 7.5.b. If we consider the unbalanced system, we are in the situation where $E[T] \to \infty$. This is because the stability condition for an individual microcell i, $\lambda_i < \mu_i$, is not fulfilled.

Simulation results and performance comparison

Figure 7.6 presents the numerical results concerning the performance of the dynamic policies in Traffic scenario 2. Figures 7.6.a and 7.6.b, on the left and right, show the simulation results when the flow size distribution is exponential and bounded Pareto, respectively. The y-axis, similar to Figure 6.2, represents

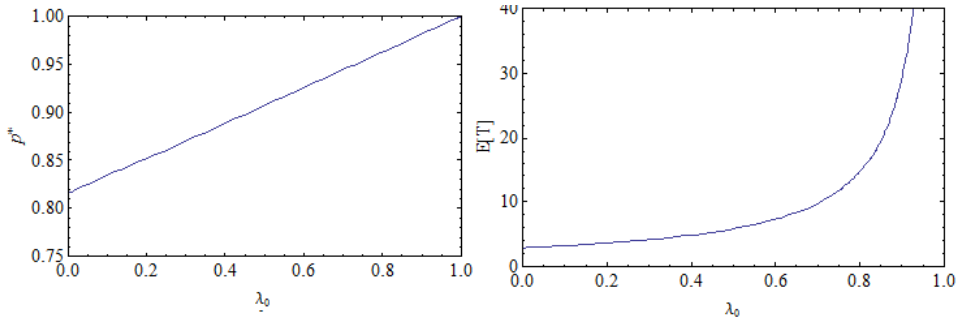

Figure 7.5: Analytical results related to the optimal static policy for Traffic scenario 2: (a) Optimal allocation probability (left) and (b) the mean sojourn time the flow spends in the system (right).

the ratio in the mean number of flows in the system between the implemented policy and the base line optimal static policy, and the x-axis shows the arrival rate to the macrocell. The curves in Figure 7.6 from top to bottom are the same as what has been mentioned in Figure 6.2. It can be observed that, similar to Traffic scenario 1, the LWL policy is not insensitive to the flow size distribution. When the flow size variance increases, there is a considerable performance degradation in the LWL policy. Even in heavily loaded traffic scenarios like this case, there is no significant difference in the performance gain between MJSQ and MP.

It can also be observed that the ratio in the mean number of flows tends to be constant for all policies, even if the arrival rate to the macrocell, λ_0, is varied. The performance gain in this traffic scenario is almost the same in all policies, except for LWL with bounded Pareto distribution and a slightly worse performance of JSQ for both flow size distributions. It can also be observed that a reduction of 35-45 % in the number of flows in the system is accomplished by using the dynamic load balancing policies.

7.4 Asymmetric scenarios

In this section, the so-called asymmetric traffic scenarios will be examined. In contrast to symmetric traffic scenarios, it is no longer assumed that the arrival

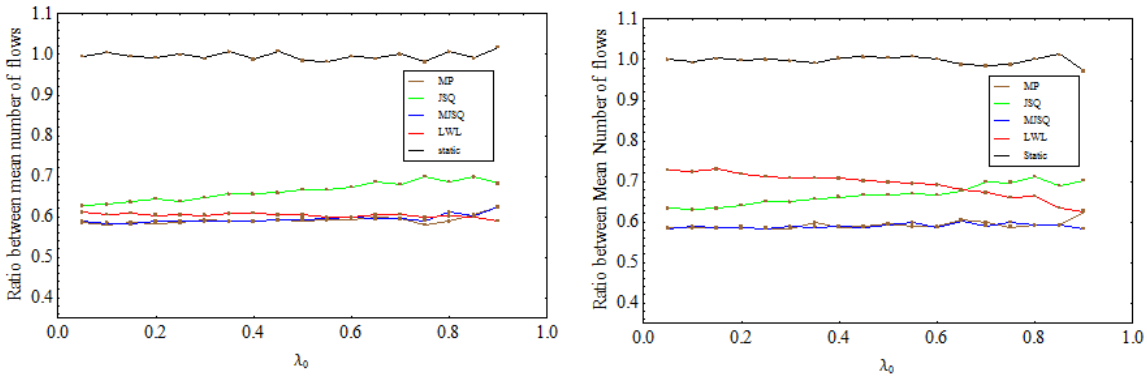

Figure 7.6: Ratio of the number of flows in the system between the dynamic and base line optimal static policies for Traffic scenario 2: (a) exponentially distributed flows (left) and (b) bounded Pareto distributed flows (right).

rate to and the service rates of the microcells in the system under consideration are equal. We will go through results obtained from two asymmetric traffic scenarios discussed in Section 7.1. Figures 7.7.a and 7.7.b illustrate Traffic scenario 3 and Traffic scenario 4, to be discussed next, respectively.

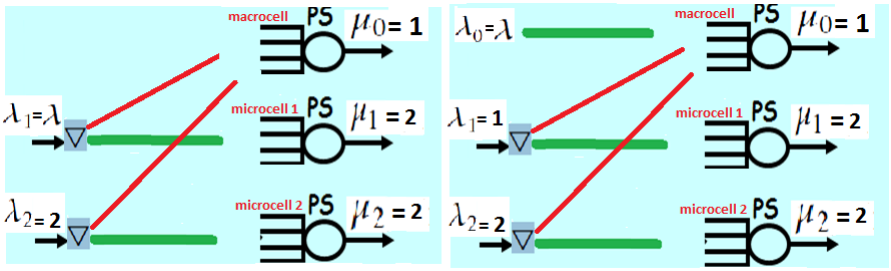

Figure 7.7: Asymmetric traffic scenarios: (left) (a) Traffic scenario 3 and (b) Traffic scanario 4 (right)

7.4.1 Traffic scenario 3

Similarly as in the symmetric traffic scenarios we consider in Traffic scenario 3 only two microcells. We assume that $\lambda_0 = 0, \lambda_2 = 2, \lambda_1 = \lambda$. So λ_1 is the free parameter to be changed in this case which is used to illustrate how the system behaves in this asymmetric situation. The service rates of the servers are the same as in Traffic scenario 1.

Stability condition and analytical results

As it is done for the symmetric scenarios above, we need to consider the stability condition to find the threshold value for λ_1. Using the stability limit calculation mentioned in (6.2), we can obtain the necessary condition for the stability of the system for the given traffic scenarios. So in this traffic scenario we have the relation $\lambda_1 < \mu_0 + \mu$. From this we obtain threshold $\lambda_1^{\max} = 3$.

Figure 7.8.a shows how the optimal allocation probability varies by changing the value of λ_1 from 0.0 to 3.0. The figure shows the effect of gradually increasing the arrival rate of the microcell 1 on the dispatching probabilities. In the case with lower arrival rate, $\lambda_1 < 1.4$, the optimal allocation probability of microcell 1, $p_1{}^*$, is 1. This means that there is no need to dispatch the arriving flows of microcell 1 to the macrocell which in turn implies that the optimal allocation probability of microcell 2, $p_2{}^*$, remains constant. However, with a higher arrival rate, $\lambda_1 > 1.4$, $p_1{}^*$ tends to decrease and $p_2{}^*$ tends to increase. This means that the need to balance the load in microcell 1 is increasing with respect to the arrival rate to the macrocell 1 after it reaches the threshold $\lambda_1 > 1.4$. However, this results in an increase in $p_2{}^*$ which reduces the rate of allocating the arriving flows of the microcell 2 to the macrocell. The figure, on the right, also shows the mean sojourn time of a flow in the system under consideration for the optimal probabilistic allocation policy.

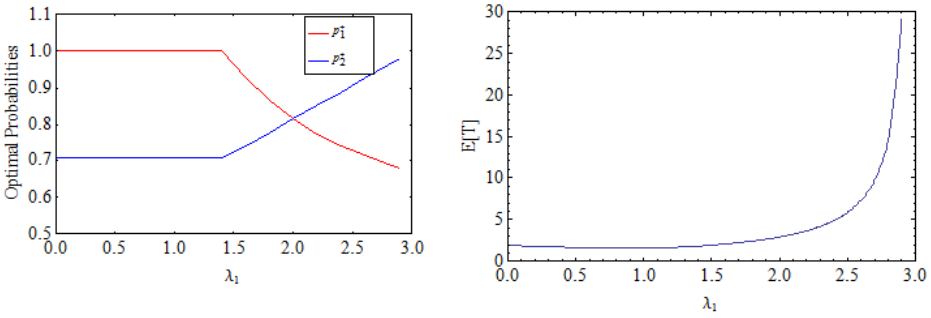

Figure 7.8: Analytical results related to the optimal static policy for Traffic scenario 3: (a) Optimal allocation probability (left) and (b) the mean sojourn time the flow spends in the system (right).

Simulation results and performance comparison

Now, we present the load balancing gain for different dynamic policies in Traffic scenario 3, as we did for the past two scenarios. Figure 7.9.a and 7.9.b show the performance gain in reducing the mean number of flow in the system by employing load balancing policies for both exponential and bounded Pareto flow size distributions. The y-axis represents the ratio in the mean number of flows in the system, as compared to the base line optimal static policy, and the x-axis shows the arrival rate to the first microcell, λ_1.

It can be observed that even in this traffic scenario, similarly as in the symmetric traffic scenarios, LWL policy is not insensitive to the flow size distribution. So we see that there is a clear performance degradation in the LWL policy for the bounded Pareto flow size distribution as compared to the exponential distribution. Figure 7.9.a reports that LWL performs slightly worse than JSQ policy in this case, unlike in other traffic scenarios studied, for lighly loaded traffic when the flow size distribution is exponential. However, in heavily loaded traffic cases LWL outperforms JSQ as usual. In addition, there is an interesting similarity in the performance gain between MJSQ and MP, i.e, they have almost the same performance.

It can also be observed that the percentage decrease in the number of flows depends on the arrival rate to the microcell, λ_1. For small values of λ_1 (lightly loaded traffic), a reduction of 30-40 % in the number of flows in the system can be achieved by using the load balancing policies. However, this percentage increases for higher arrival rates. For larger values of λ_1, a reduction of 40-50 % can be achieved.

7.4.2 Traffic scenario 4

Traffic scenario 4 is quite similar to Traffic scenario 2 with the only exception that the arrival rates to the micro-cells are asymmetric. We assume that we have two microcells with the arrival rates $\lambda_1 = 1$ and $\lambda_2 = 2$, and a single macrocell with a variable arrival rate of $\lambda_0 = \lambda$. So λ_0 is the free parameter to be changed in this case. The service rates of the servers are the same as in

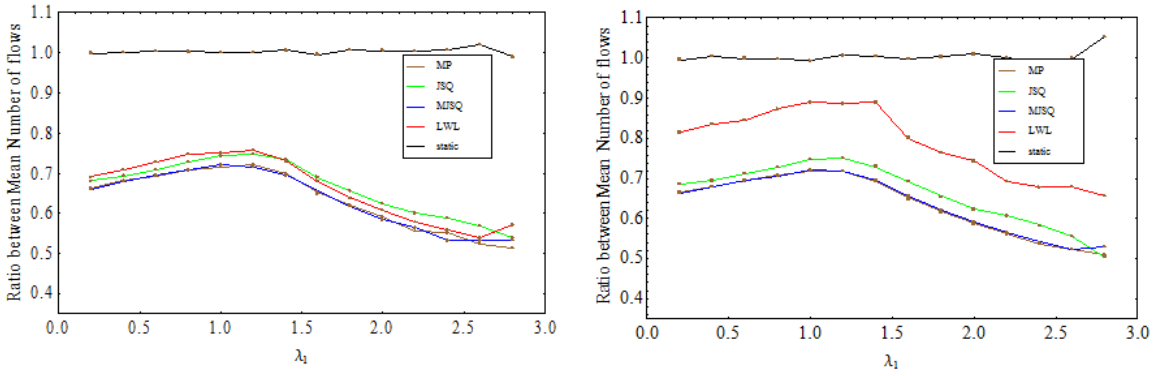

Figure 7.9: Ratio of the number of flows in the system between the dynamic and base line optimal static policies for Traffic scenario 3: (a) exponentially distributed flows (left) and (b) bounded Pareto distributed flows (right).

Traffic scenario 1.

Stability condition and analytical results

Similarly as in Traffic scenario 2 the stability condition for this traffic scenario is determined solely by the arrival rate of the macrocell. Thus, the threshold for stability is $\lambda_0^{\max} = 1$.

Figure 7.10.a shows the optimal allocation probability when the arriving traffic rate in the macrocell, λ_0, varies from 0 to 1. It can be observed that there is no need to dispatch the arriving flow of microcell 1 to the macrocell as the probability is 1 for all traffic conditions in the macrocell. This is because microcell 1 is lightly loaded, and it is not recommended to dispatch the load to the server with smaller service rate. However, this is not the case for the arriving flows in microcell 2. In contrast to microcell 1, the load in microcell 2 is high and it is recommended to dispatch the arriving flows to the macrocell with the given probability. It can also be observed that increasing the dedicated arrival rate λ_0 results in an increase in the optimal allocation probability of microcell 2, $p_2{}^*$, which in turn reduces the probability of switching the arrival from microcell 2 to the macrocell. The mean sojourn time of a flow in the system for the optimal probabilistic allocation policy is shown in Figure 7.10.b

.

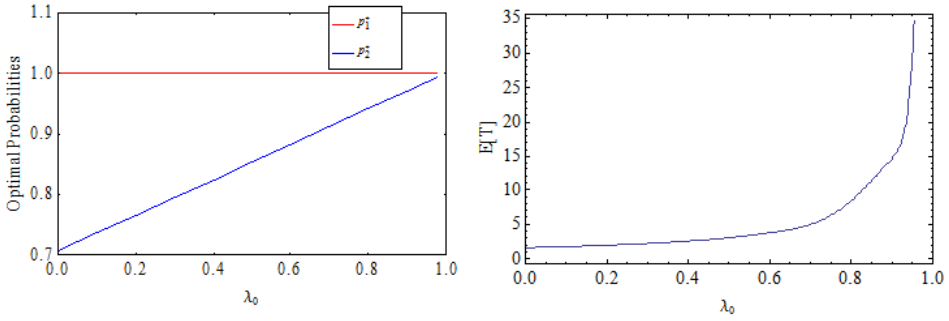

Figure 7.10: Analytical results related to the optimal static policy for Traffic scenario 4: (a) Optimal allocation probability (left) and (b) the mean sojourn time the flow spends in the system (right).

Simulation results and performance comparison

Figure 7.11 presents the numerical results concerning the performance of dynamic policies in Traffic scenario 4. Figures 6.8.a and 6.8.b, on the left and right, show the simulation results when the flow size distribution is exponential and bounded Pareto, respectively. The y-axis represents the ratio in the mean number of flows in the system, as compared to the base line optimal static policy, and the x-axis shows the arrival rate to the macrocell. The curves in Figure 7.11 from top to bottom are the same as already mentioned in Figure 7.4. It can be observed that, once again, LWL is not insensitive to the flow size distribution. When the flow size variance increases, there is a slight performance degradation for the LWL policy. Even for asymmetric and partly loaded traffic scenarios like this case, there is no significant difference between the performance gain between MJSQ and MP.

It can also be observed that, similar to Traffic scenario 2, the percentage decrease in the number of flows tends to be constant even if the arrival rate to the macrocell, λ_0, is varied. The performance gain is almost the same in all policies, except LWL performs worst for the bounded Pareto distribution, with this traffic scenario. It can also be observed that a reduction of 25-30 % in the number of flows in the system is accomplished by using the dynamic load balancing policies.

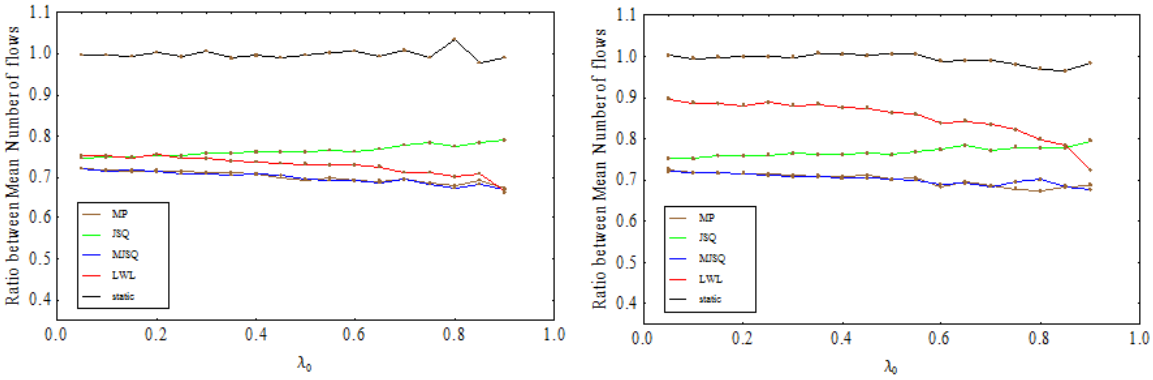

Figure 7.11: Ratio of the number of flows in the system between the dynamic and base line optimal static policies for Traffic scenario 4: (a) exponentially distributed flows (left) and (b) bounded Pareto distributed flows (right).

7.5 Effect of the flow size variation

The purpose of this section is to investigate what happens in the performance of the load balancing policies when changing the variance of the flow size distribution. Traffic scenario 2 is simulated for bounded Pareto distribution by changing the shape parameter α. As discussed in Section 3.1, the lower the α parameter, the more variable the flow size distribution will be. The results of the simulations are presented in Figure 7.12.

It can be seen from the figure that the performance of almost all policies, except LWL, is insensitive with respect to the flow size distribution. As discussed in the previous sections, LWL performs poorly when the flow size variation is high. Although it requires extra information, it performs even worse than JSQ for the shape parameter $\alpha = 1.5$ and $\alpha = 2.0$. However, when $\alpha = 3.0$, the performance of the LWL policy resembles that of the LWL policy with exponential flow size distribution. Two main reasons were mentioned in [23] for the performance degradation of LWL for the highly varying distributions. The first one is related to the scheduling policy used, i.e, processor sharing discipline. For PS systems the unfinished work does not capture the delaying effect of the incoming flow. The other one is related to the service rate of the servers. The LWL policy does not take into account the differences in these service rates as regards the arriving flow.

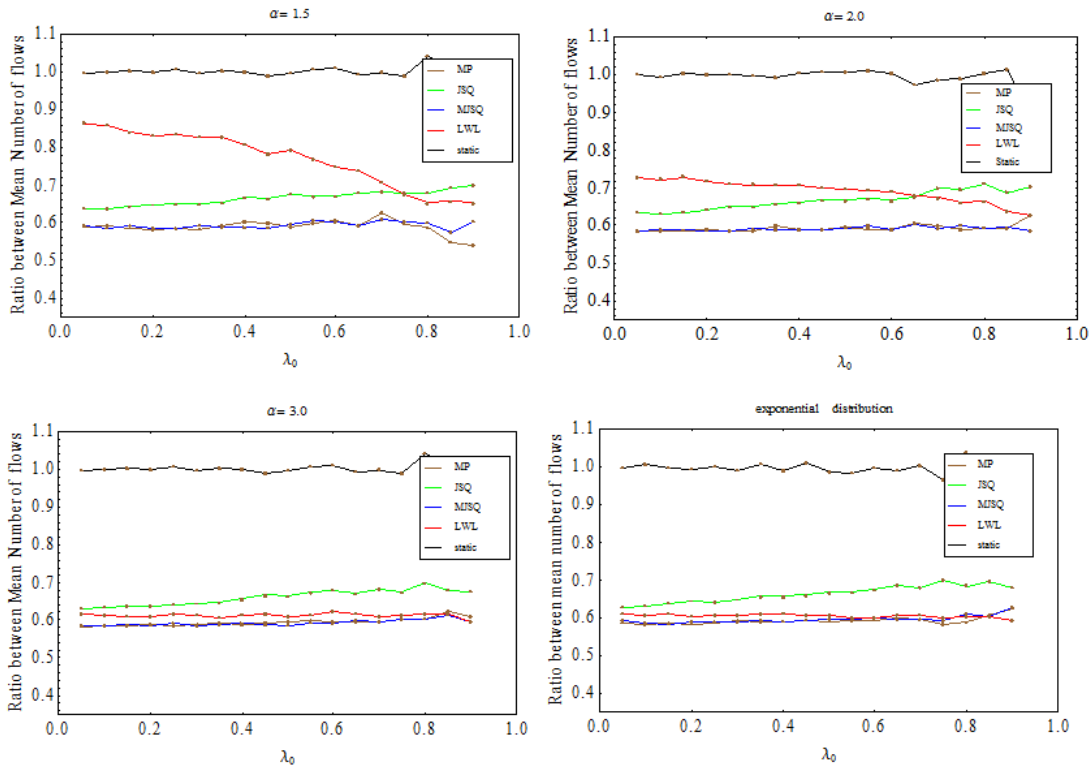

Figure 7.12: Illustration of effect of the flow size variation: (a) bounded Pareto flow size distribution: $\alpha = 1.5$ (top left), (b) bounded Pareto flow size distribution: $\alpha = 2.0$. (top right), (c) bounded Pareto flow size distribution: $\alpha = 3.0$ (bottom left) and (d) exponential flow size distribution (bottom right)

7.6 Effect of the number of microcells

The effect of increasing the number of microcells, n, that are within a single macro-cell is studied in this section by simulations. We consider a symmetric case where the arrival rate to the microcells is assumed to be the same. In these simulations we assume $\lambda_0 = \lambda, \lambda_1 = \lambda_2 = \cdots = \lambda_n = 2$ and $\mu_0 = 1, \mu_1 = \mu_2 = \cdots = \mu_n = 2$. The simulation is done for both the exponential and bounded Pareto distributions.

Figure 7.13 shows the impact of the number of micro-cells on the percentage performance gain of the load balancing policies. The left hand side of the figure is for the exponential flow size distribution and the right hand side is for the bounded Pareto distribution. The y-axis represents the percentage decrease in the number of flows in the system, as compared to the base line optimal static policy, and the x-axis shows the number of microcells in the system. There is around 35 % performance gain, compared to the optimal static policies, with respect to the number of users in the system when we consider two microcells. However, this percentage increases when we increase the number of microcells. It will reach around 50 % when the number of microcells becomes 10. Similar to other traffic scenarios studied in prevouis sections, the variation in the performance of the MJSQ and MP policies is small.

It can also be observed that, for the bounded Pareto distribution, the performance of the LWL policy seems to be slightly worse than other dynamic policies. With smaller arrival rates to the microcells, the LWL policy performs even worse than the JSQ policy. However, with higher values of λ_0, LWL outperforms the JSQ policy.

7.7 Summary of the results

All implemented dynamic policies seem to improve the performance of the system under consideration, with respect to the mean delay, compared to the optimal static policy in all traffic scenarios which have been examined. The

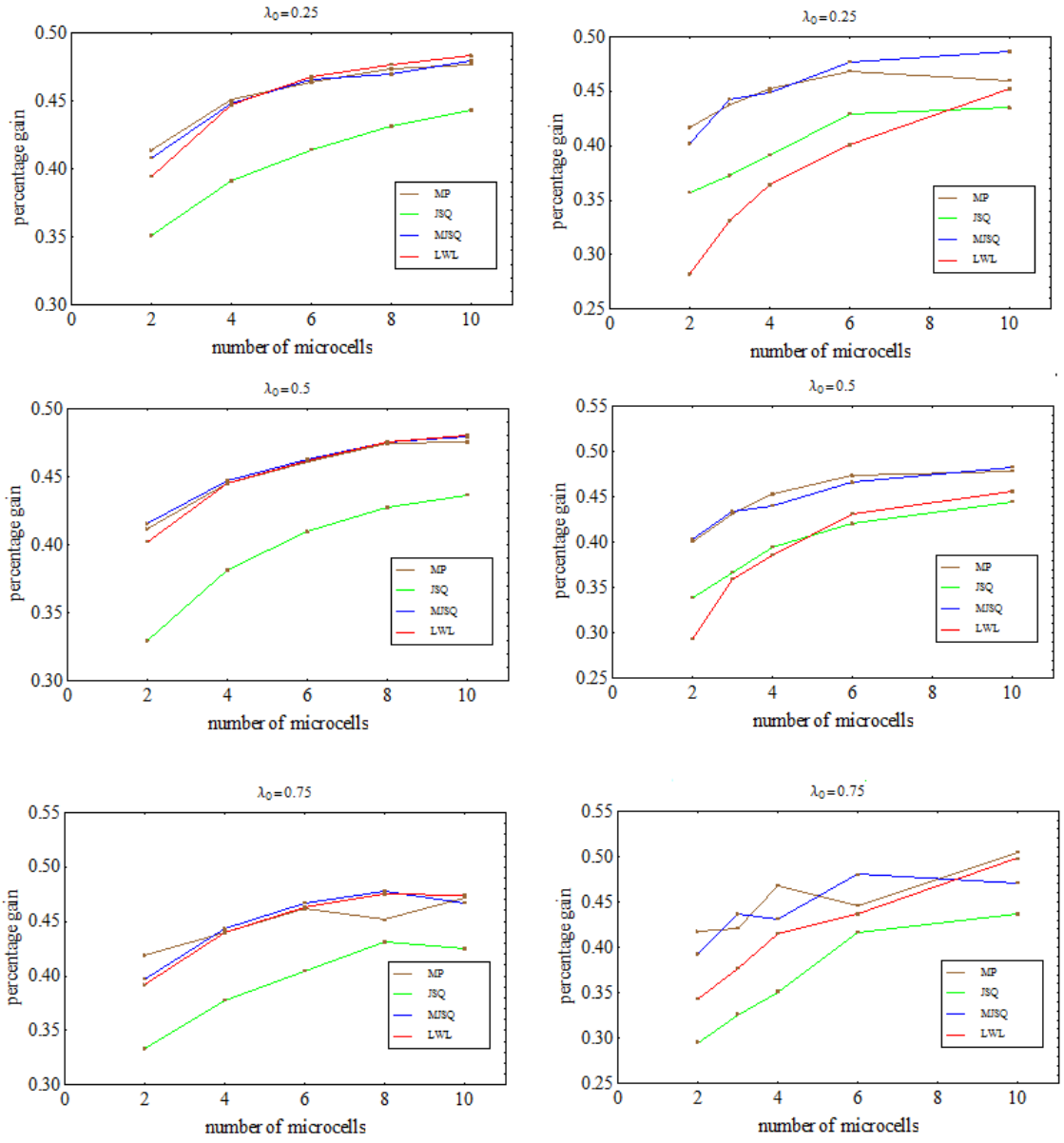

Figure 7.13: Impact of the number of micro-cells on the performance gain of load balancing policies: (left) the exponential flow size distribution and (right) the bounded Pareto flow size distribution.

improvement depends on the traffic load in the system. The performance gain appears to be higher with highly loaded traffic scenarios than with lightly loaded ones. We found that MJSQ and MP perform better than JSQ and LWL. In addition, the performance gain between MSQ and MP was very similar. This is an interesting observation because MP needs additional information compared to MJSQ.

We examined how the dynamic policies behave for highly varying distributions. We found out that all implemented policies, except LWL, are nearly insensitive to the distribution of flow sizes. Moreover, we also considered the effect of increasing the number of microcells on the performance gain. We found out that increasing the number of microcells results in an increase in the performance gain of the system.

Chapter 8

Conclusions

8.1 Summary

The purpose of this thesis was to investigate the problem of load balancing of elastic data traffic in heterogeneous networks. We have modelled a heterogeneous network consisting of a single macrocell and a number of microcells as a distributed server system employing a processor sharing discipline at each server. Each cell has been modelled as an $M/G/1-PS$ queueing system.

Both static and dynamic load balancing policies have been studied. The static policy corresponds to the probabilistic allocation policy. The problem of finding optimal allocation probabilities, the load balancing problem, was formulated as a mathematical optimization problem with the objective of minimizing the mean flow delay of the system. For symmetric traffic scenarios, the explicit solution for the optimal allocation probability was found. However, for asymmetric traffic scenarios we argued that the optimization problem at hand is a standard convex optimization problem that can be numerically solved.

For the dynamic case, four state dependent policies were implemented to study the performance of the dynamic load balancing is used. The policies are JSQ, LWL, MJSQ and MP. An event-driven simulator was developed for each dynamic policy. To compare the performance of different policies, the mean number of flows in the system was used as a performance metric.

The performance of the implemented policies both in symmetric and asymmetric traffic scenarios at the flow level was explored. It was found that all dynamic policies can significantly improve the flow level delay performance in the system under consideration compared with the optimal static policy. We also found out that MJSQ and MP seem to be better policies than the other implemented policies. Although MJSQ needs much less information, in all traffic scenarios studied, it was observed that there is not much difference between the MJSQ and MP policies in their performance gain with respect to the optimal static policy. Additionally, the results indicated that the highest performance improvement is achieved when the load of the system is high.

The effect of increasing the number of microcells on the performance of dynamic policies was also studied. It was shown that increasing the number of microcells also increases the gain from load balancing. In addition, the impact of service rate difference between macrocell and microcells on the performance gain of dynamic policies has been studied in Appendix A.

It has also been demonstrated how the use of the remaining workload present at each server in the LWL policy could lead to a relatively poor load balancing policy for PS systems, while it is good for FIFO systems. One of the interesting findings that has been noticed was the approximate insensitivity of the average response time under the implemented dynamic policies, except LWL, to the variance of the flow size distribution. This result suggests that the insensitivity property of a single server PS system might also apply, at least approximately, for the more complex distributed server system considered here.

8.2 Future work

For future research, this study can be extended in several directions. First, in this thesis our assumption in modelling the system is that we only consider elastic flows. However, an examination of mobile data by application shows that video is the greatest contributor in mobile data traffic. This multimedia traffic consists of UDP (streaming) flows. So, streaming flows constitute a larger portion of the mobile data traffic. Thus, studying the behaviour of the

system with the assumption that the arrival process consists of both elastic and streaming flows is more realistic.

Another direction for future work is in generalizing the basic system model used in this thesis to take into account real world situations. For example, the underlying assumption in this study was that the service rate of microcells is larger than that of the macrocell. However, a radio model was not used to actually determine the service rates of the users in the microcells and the macrocell.

Third, in this thesis the performance of the implemented dynamic policies are studied with a single metric, i.e., with the average flow level delay. Additional metrics, such as fairness, should be studied since they provide greater insight into the performance of the system.

The fourth direction of future work is to study if it is possible to optimize the implemented dynamic policies. Although four dynamic load balancing policies are implemented and quite good performance gain is achieved with these policies, none of the policies is optimal. The Markov Decision Process (MDP) framework can be used to determine an efficient and robust state dependent policy. With the implemented dynamic polices at hand, carrying out the first policy iteration (FPI) may yield improved state dependent policies.

Bibliography

[1] H. Holma and A. Toskala, *LTE for UMTS Evolution to LTE-Advanced*. Wiley, 2011.

[2] A. Agrawal, "Heterogeneous networks: A new paradigm for increasing cellular capacity," Qualcomm, Tech. Rep., 2009.

[3] "Traffic management and offload strategies for operators," Qualcomm, Tech. Rep., January 2011. [Online]. Available: http://www.qualcomm.com/media/documents/traffic-management-and-offload-strategies-operators

[4] "Femtocells - natural solution for offload," Femto Forum, Tech. Rep., June 2010. [Online]. Available: http://lteworld.org/whitepaper/femtocells-%E2%80%93-natural-solution-offload

[5] L. Kleinrock, *Queueing Systems, Volume 1, Theory*. Wiley-Interscience, 1975.

[6] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *SIGMETRICS Perform. Eval. Rev.*, vol. 24, no. 1, pp. 160–169, May 1996. [Online]. Available: http://doi.acm.org/10.1145/233008.233038

[7] G. Irlam, "Unix file size survey," 1994, availabe in the web at http://www.gordoni.com/ufs93.html.

[8] M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," *J. Parallel*

Distrib. Comput., vol. 59, no. 2, pp. 204–228, Nov. 1999. [Online]. Available: http://dx.doi.org/10.1006/jpdc.1999.1577

[9] B. Schroeder and M. Harchol-Balter, "Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness," *Cluster Computing*, vol. 7, no. 2, pp. 151–161, Apr. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:CLUS.0000018564.05723.a2

[10] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, 1953. [Online]. Available: http://dx.doi.org/10.2307/2236285

[11] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 687–690, 1968.

[12] J. Roberts, "Traffic theory and the internet," *Communications Magazine, IEEE*, vol. 39, no. 1, pp. 94 –99, Jan 2001.

[13] V. B. Iversen, *Teletrafffic Engineering and Network Planning*. Technical University of Denmark, 2010.

[14] T. Bonald and J. W. Roberts, "Congestion at flow level and the impact of user behaviour," *Comput. Netw.*, vol. 42, no. 4, pp. 521–536, Jul. 2003. [Online]. Available: http://dx.doi.org/10.1016/S1389-1286(03)00200-7

[15] T. Bonald and A. Proutière, "Wireless downlink data channels: user performance and cell dimensioning," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 339–352. [Online]. Available: http://doi.acm.org/10.1145/938985.939020

[16] S. B. Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts, "Statistical bandwidth sharing: a study of congestion at flow level," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*,

ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 111–122. [Online]. Available: http://doi.acm.org/10.1145/383059.383068

[17] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Pearson Higher Ed USA, 2010.

[18] C. Nuzman, I. Saniee, W. Sweldens, and A. Weiss, "A compound model for TCP connection arrivals for lan and wan applications," *Comput. Netw.*, vol. 40, no. 3, pp. 319–337, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1016/S1389-1286(02)00298-0

[19] E. Hyytiä, S. Aalto, and A. Penttinen, "Minimizing slowdown in heterogeneous size-aware dispatching systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 29–40, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2318857.2254763

[20] E. Hyytiä, J. Virtamo, S. Aalto, and A. Penttinen, "M/M/1-PS queue and size-aware task assignment," *Perform. Eval.*, vol. 68, no. 11, pp. 1136–1148, Nov. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2011.07.011

[21] M. Harchol-Balter, "Task assignment with unknown duration," *J. ACM*, vol. 49, no. 2, pp. 260–288, Mar. 2002. [Online]. Available: http://doi.acm.org/10.1145/506147.506154

[22] F. Bonomi, "On job assignment for a parallel system of processor sharing queues," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 858–869, Jul. 1990. [Online]. Available: http://dx.doi.org/10.1109/12.55688

[23] Z. Tari, J. Broberg, A. Y. Zomaya, and R. Baldoni, "A least flow-time first load sharing approach for distributed server farm," *J. Parallel Distrib. Comput.*, vol. 65, no. 7, pp. 832–842, Jul. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2005.02.007

[24] W. Winston, "Optimality of the shortest line discipline," vol. 14, no. 7, pp. 181–189, 1977.

[25] J. Lui, R. R. Muntz, and D. Towsley, "Bounding the mean response time of the minimum expected delay routing policy: An algorithmic approach," *IEEE Trans. Comp*, vol. 44, pp. 1371–1382, 1995.

[26] H. Feng, V. Misra, and D. Rubenstein, "Optimal state-free, size-aware dispatching for heterogeneous M/G-type systems," *Performance Evaluation*, vol. 62, no. 1–4, pp. 475 – 492, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531605001100

Appendix A

Other traffic and system scenarios

The impact of service rate difference between macrocell and microcells on the performance gain of dynamic policies has been studied in this appendix. The same four different traffic and system scenarios have been considered as earlier. The only difference in this case is in the capacity of the servers. In the previous traffic scenarios, studied in Section 7.3 and 7.4, we specified the mean service rates of microcells as $\mu_1 = \mu_2 = 2$. However, in this case we assume the mean service rates of the servers as $\mu_1 = \mu_2 = 4$. The results obtained from studying each of the traffic and system scenarios are shown in the figures below. In all traffic scenarios, we assume that flows arrive to each server i as a Poisson stream with rate λ_i and the flow sizes follow an exponential distribution.

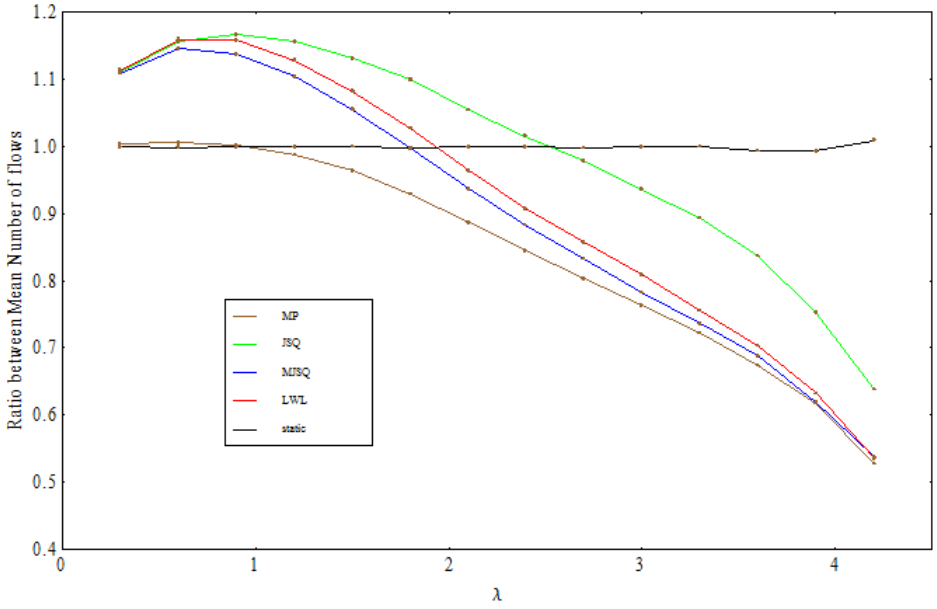

Figure A.1: Traffic scenario 1

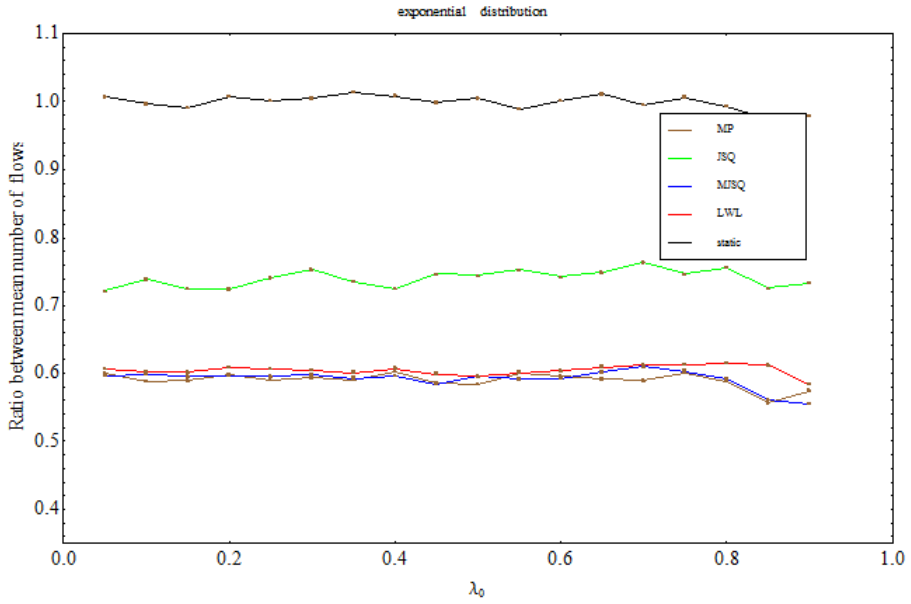

Figure A.2: Traffic scenario 2

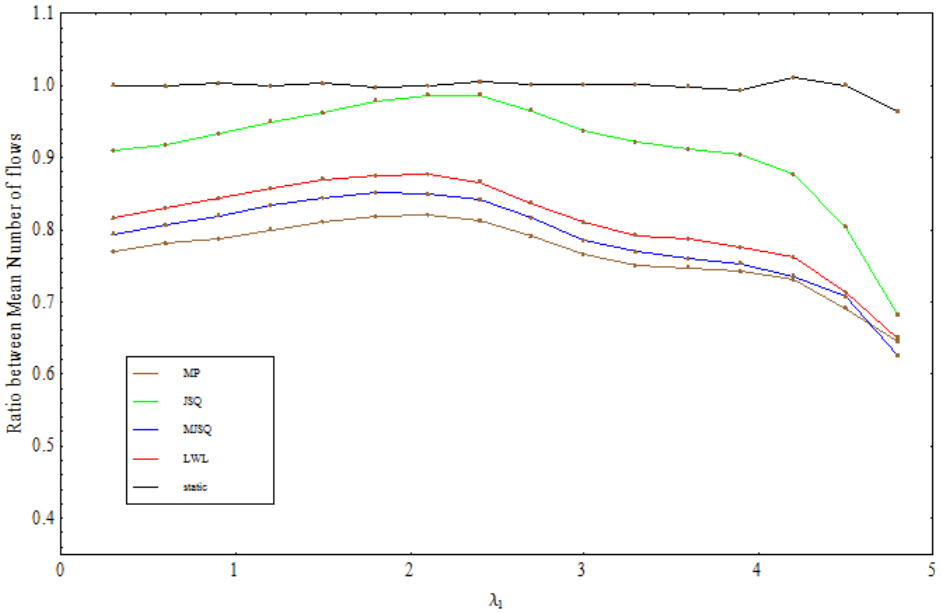

Figure A.3: Traffic scenario 3

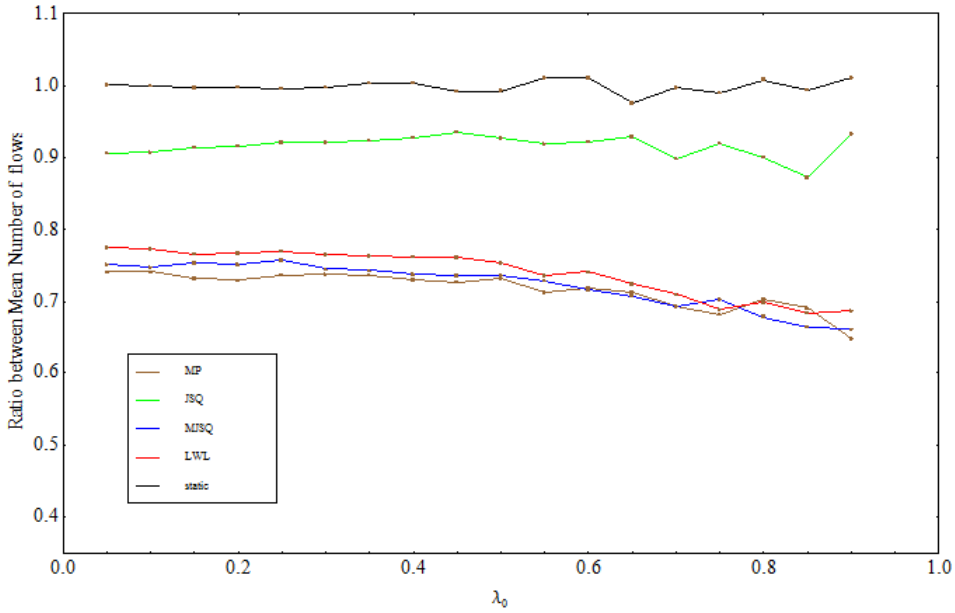

Figure A.4: Traffic scenario 4

Appendix B

Mathematica source codes of the simulator for the implemented policies .

1. OPTIMAL STATIC

```
StaticSimilatorfullv1[lambda_, mu_, final_] :=
 Module[{x, tcurr, tprev, ta, td, que, counter, i, list,
   listdepart, index, xx, xtotal, p, mu0, lambda0, n, jobsize},
  xtotal = 0;
  mu0 = mu[[1]];
  lambda0 = lambda[[1]];
  n = Length[lambda] - 1;
  ta = Table[0, {Length[lambda]}];
  td = Table[0, {Length[lambda]}];
  xx = Table[0, {Length[lambda]}];
  For[i = 1, i <= Length[lambda],
   i++,
   If[ lambda[[i]] > 0,
    ta[[i]] = RandomVariate[ExponentialDistribution[lambda[[i]]]],
    ta[[i]] = Infinity]];
  For[i = 1, i <= Length[lambda],
   i++, td[[i]] = Infinity];
  x = Table[0, {Length[lambda]}];
  tcurr = Table[0, {Length[lambda]}];
  tprev = Table[0, {Length[lambda]}];
  que = {};
  que = Table[que, {Length[lambda]}];
  counter = 1;
  list = Table[0, {Length[lambda]}];
  listdepart = Table[0, {Length[lambda]}];
```

```
While[counter ≤ final,
 If[Min[ta] < Min[td],
  {
   For[i = 1, i <= Length[lambda],
    i++, list[[i]] = {ta[[i]], i}];
   list = Sort[list];
   jobsize = RandomVariate[ExponentialDistribution[1]];
   If[ta[[1]] == list[[1, 1]],
    {
     i = 1;
     tprev[[i]] = tcurr[[i]];
     tcurr[[i]] = ta[[i]];
     xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
     If[ lambda[[i]] > 0,
      ta[[i]] = tcurr[[i]] + RandomVariate[
          ExponentialDistribution[lambda[[i]]]], ta[[i]] = Infinity];
    },
    {
     index = list[[1, 2]];
     p = pro[mu0, mu[[index]], lambda0, lambda[[index]], n];
     i = StaticDispatcher1[p, index];
     tprev[[i]] = tcurr[[i]];
     tcurr[[i]] = list[[1, 1]];
     xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
     ta[[index]] =
      tcurr[[i]] + RandomVariate[ExponentialDistribution[lambda[[index]]]]
    }];
   If[Length[que[[i]]] > 0,
    que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]])];
   que[[i]] = Append[que[[i]], jobsize / mu[[i]]];
   que[[i]] = Sort[que[[i]]];
   x[[i]] = x[[i]] + 1;
   counter = counter + 1;
    td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]]
  },
  {
```

```
For[i = 1, i <= Length[lambda],
         i++, listdepart[[i]] = {td[[i]], i}];
       listdepart = Sort[listdepart];
       i = listdepart[[1, 2]];
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = listdepart[[1, 1]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       x[[i]] = x[[i]] - 1;
       If[Length[que[[i]]] > 0,
         que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]]);
         que[[i]] = Delete[que[[i]], 1];
        ];
       If[x[[i]] > 0, td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]],
         td[[i]] = Infinity];

       }
      ]];
  For[i = 1, i <= Length[lambda],
    i++,
    If[tcurr[[i]] > 0,
     xx[[i]] = xx[[i]] / tcurr[[i]]];
    xtotal = xtotal + xx[[i]]
   ];
  xtotal

  ];
StaticDispatcher1[p_, index_] :=
 Module[{a},
  a = RandomReal[];
  If[p >= 1, Return[index],
   If[p > a, Return[index],
     Return[1]
    ];]
 ]
```

2. JOIN THE SHORTEST QUEUE

```
DynamicSimilatorfullv1[lambda_, mu_, final_] :=
 Module[{x, tcurr, tprev, ta, td, que,
   counter, i, list, listdepart, index, xx, xtotal, jobsize},
  xtotal = 0;
  ta = Table[0, {Length[lambda]}];
  td = Table[0, {Length[lambda]}];
  xx = Table[0, {Length[lambda]}];
  For[i = 1, i <= Length[lambda],
   i++,
   If[ lambda[[i]] > 0,
    ta[[i]] = RandomVariate[ExponentialDistribution[lambda[[i]]]],
    ta[[i]] = Infinity]];
  For[i = 1, i <= Length[lambda],
   i++, td[[i]] = Infinity];
  x = Table[0, {Length[lambda]}];
  tcurr = Table[0, {Length[lambda]}];
  tprev = Table[0, {Length[lambda]}];
  que = {};
  que = Table[que, {Length[lambda]}];
  counter = 1;
  list = Table[0, {Length[lambda]}];
  listdepart = Table[0, {Length[lambda]}];
  While[counter ≤ final,
   If[Min[ta] < Min[td],
    {
     For[i = 1, i <= Length[lambda],
      i++, list[[i]] = {ta[[i]], i}];
     list = Sort[list];
     jobsize = RandomVariate[ExponentialDistribution[1]];
     If[ta[[1]] == list[[1, 1]],
      {
       i = 1;
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = ta[[i]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       If[ lambda[[i]] > 0,
        ta[[i]] = tcurr[[i]] + RandomVariate[
            ExponentialDistribution[lambda[[i]]]], ta[[i]] = Infinity];
      },
```

```
{
          index = list[[1, 2]];
          jobsize = RandomVariate[ExponentialDistribution[1]];
          i = DynamicDispatcher1[x, index];
          tprev[[i]] = tcurr[[i]];
          tcurr[[i]] = list[[1, 1]];
          xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
          ta[[index]] =
           tcurr[[i]] + RandomVariate[ExponentialDistribution[lambda[[index]]]]
         }
         ];
       If[Length[que[[i]]] > 0,
         que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]])];
       que[[i]] = Append[que[[i]], jobsize / mu[[i]]];
       que[[i]] = Sort[que[[i]]];
       x[[i]] = x[[i]] + 1;
       counter = counter + 1;
       td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]]
       },
      {
       For[i = 1, i <= Length[lambda],
        i++, listdepart[[i]] = {td[[i]], i}];
       listdepart = Sort[listdepart];
       i = listdepart[[1, 2]];
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = listdepart[[1, 1]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       x[[i]] = x[[i]] - 1;
       If[Length[que[[i]]] > 0,
        que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]]);
        que[[i]] = Delete[que[[i]], 1];
       ];
        If[x[[i]] > 0, td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]],
         td[[i]] = Infinity];
      }
      ]];
  For[i = 1, i <= Length[lambda],
    i++,
    xx[[i]] = xx[[i]] / tcurr[[i]];
    xtotal = xtotal + xx[[i]]
   ];
  xtotal

  ];
DynamicDispatcher1[x_, index_] :=
 Module[{a},
  If[x[[index]] <= x[[1]], Return[index],
    Return[1]
   ];
 ]
```

3. LEAST WORK LOAD

```
DynamicSimilatorfullvworkload[lambda_, mu_, final_] :=
 Module[{x, tcurr, tprev, ta, td, que, counter,
   i, list, listdepart, index, quesum, xx, xtotal, j, jobsize},
  xtotal = 0;
  ta = Table[0, {Length[lambda]}];
  quesum = Table[0, {Length[lambda]}];
  td = Table[0, {Length[lambda]}];
  xx = Table[0, {Length[lambda]}];
  For[i = 1, i <= Length[lambda],
   i++,
   If[ lambda[[i]] > 0,
    ta[[i]] = RandomVariate[ExponentialDistribution[lambda[[i]]]],
    ta[[i]] = Infinity]];
  For[i = 1, i <= Length[lambda],
   i++, td[[i]] = Infinity];
  x = Table[0, {Length[lambda]}];
  tcurr = Table[0, {Length[lambda]}];
  tprev = Table[0, {Length[lambda]}];
  que = {};
  que = Table[que, {Length[lambda]}];
  counter = 1;
  list = Table[0, {Length[lambda]}];
  listdepart = Table[0, {Length[lambda]}];
  While[counter ≤ final,
   If[Min[ta] < Min[td],
    {
     For[i = 1, i <= Length[lambda],
      i++, list[[i]] = {ta[[i]], i}];
     list = Sort[list];
     jobsize = RandomVariate[ExponentialDistribution[1]];
     If[ta[[1]] == list[[1, 1]],
      {
       i = 1;
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = ta[[i]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       If[ lambda[[i]] > 0,
        ta[[i]] = tcurr[[i]] + RandomVariate[
            ExponentialDistribution[lambda[[i]]]], ta[[i]] = Infinity];
      },
```

```
{
          index = list[[1, 2]];
          quesum[[index]] = Sum[que[[index, j]], {j, 1, Length[que[[index]]]}];
          quesum[[1]] = Sum[que[[1, j]], {j, 1, Length[que[[1]]]}];
          i = DynamicDispatcher1[x, quesum, index];
          tprev[[i]] = tcurr[[i]];
          tcurr[[i]] = list[[1, 1]];
          xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
          ta[[index]] =
           tcurr[[i]] + RandomVariate[ExponentialDistribution[lambda[[index]]]]
         }
         ];
       If[Length[que[[i]]] > 0,
         que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]])];
       que[[i]] = Append[que[[i]], jobsize / mu[[i]]];
       que[[i]] = Sort[que[[i]]];
       x[[i]] = x[[i]] + 1;
       counter = counter + 1;
       td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]]
       },
      {
       For[i = 1, i <= Length[lambda],
        i++, listdepart[[i]] = {td[[i]], i}];
       listdepart = Sort[listdepart];
       i = listdepart[[1, 2]];
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = listdepart[[1, 1]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       x[[i]] = x[[i]] - 1;
       If[Length[que[[i]]] > 0,
        que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]]);
        que[[i]] = Delete[que[[i]], 1];
       ];
       If[x[[i]] > 0, td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]],
        td[[i]] = Infinity];
      }
      ]];
  For[i = 1, i <= Length[lambda],
    i++,
    xx[[i]] = xx[[i]] / tcurr[[i]];
    xtotal = xtotal + xx[[i]]
   ];
  xtotal
  ];
DynamicDispatcher1[x_, quesum_, index_] :=
 Module[{a},
  If[quesum[[index]] <= quesum[[1]], Return[index],
    Return[1]
   ];
 ]
```

4. MODIFIED JOIN THE SHORTEST QUEUE

```
DynamicSimilatorfullvservicerate[lambda_, mu_, final_] :=
 Module[{x, tcurr, tprev, ta, td, que,
   counter, i, list, listdepart, index, xx, xtotal, jobsize},
  xtotal = 0;
  ta = Table[0, {Length[lambda]}];
  td = Table[0, {Length[lambda]}];
  xx = Table[0, {Length[lambda]}];
  For[i = 1, i <= Length[lambda],
   i++,
   If[ lambda[[i]] > 0,
    ta[[i]] = RandomVariate[ExponentialDistribution[lambda[[i]]]],
    ta[[i]] = Infinity]];
  For[i = 1, i <= Length[lambda],
   i++, td[[i]] = Infinity];
  x = Table[0, {Length[lambda]}];
  tcurr = Table[0, {Length[lambda]}];
  tprev = Table[0, {Length[lambda]}];
  que = {};
  que = Table[que, {Length[lambda]}];
  counter = 1;
  list = Table[0, {Length[lambda]}];
  listdepart = Table[0, {Length[lambda]}];
  While[counter ≤ final,
   If[Min[ta] < Min[td],
    {
     For[i = 1, i <= Length[lambda],
      i++, list[[i]] = {ta[[i]], i}];
     list = Sort[list];
     jobsize = RandomVariate[ExponentialDistribution[1]];
     If[ta[[1]] == list[[1, 1]],
      {
       i = 1;
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = ta[[i]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       If[ lambda[[i]] > 0,
        ta[[i]] = tcurr[[i]] + RandomVariate[
            ExponentialDistribution[lambda[[i]]]], ta[[i]] = Infinity];
      },
```

```
{
          index = list[[1, 2]];
          i = DynamicDispatcher1[x, mu, index];
          tprev[[i]] = tcurr[[i]];
          tcurr[[i]] = list[[1, 1]];
          xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
          ta[[index]] =
           tcurr[[i]] + RandomVariate[ExponentialDistribution[lambda[[index]]]]
         }
         ];
       If[Length[que[[i]]] > 0,
         que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]])];
       que[[i]] = Append[que[[i]], jobsize/mu[[i]]];
       que[[i]] = Sort[que[[i]]];
       x[[i]] = x[[i]] + 1;
       counter = counter + 1;
       td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]]
       },
      {
       For[i = 1, i <= Length[lambda],
        i++, listdepart[[i]] = {td[[i]], i}];
       listdepart = Sort[listdepart];
       i = listdepart[[1, 2]];
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = listdepart[[1, 1]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       x[[i]] = x[[i]] - 1;
       If[Length[que[[i]]] > 0,
        que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]]);
        que[[i]] = Delete[que[[i]], 1];
       ];
        If[x[[i]] > 0, td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]],
         td[[i]] = Infinity];
      }
      ]];
  For[i = 1, i <= Length[lambda],
    i++,
    xx[[i]] = xx[[i]] / tcurr[[i]];
    xtotal = xtotal + xx[[i]]
   ];
  xtotal
  ];
DynamicDispatcher1[x_, mu_, index_] :=
 Module[{a, b},
  a = x[[index]] / mu[[index]];
  b = x[[1]] / mu[[1]];
  If[a ≤ b, Return[index],
   Return[1]
  ];
 ]
```

5. MYOPIC

```
DynamicSimilatorfullmyopic[lambda_, mu_, final_] :=
 Module[{x, tcurr, tprev, ta, td, que, quecus, counter, i, list, listdepart,
   index, delaysubtraction, xx, xtotal, j, m, newmu, n, serv, comparserv},
  xtotal = 0;
  newmu = Table[0, {Length[lambda]}];
  quecus = Table[0, {Length[lambda]}];
  ta = Table[0, {Length[lambda]}];
  delaysubtraction = Table[0, {Length[lambda]}];
  td = Table[0, {Length[lambda]}];
  xx = Table[0, {Length[lambda]}];
  For[i = 1, i <= Length[lambda],
   i++,
   If[ lambda[[i]] > 0,
    ta[[i]] = RandomVariate[ExponentialDistribution[lambda[[i]]]],
    ta[[i]] = Infinity]];
  For[i = 1, i <= Length[lambda],
   i++, td[[i]] = Infinity];
  x = Table[0, {Length[lambda]}];
  tcurr = Table[0, {Length[lambda]}];
  tprev = Table[0, {Length[lambda]}];
  que = {};
  que = Table[que, {Length[lambda]}];
  counter = 1;
  list = Table[0, {Length[lambda]}];
  listdepart = Table[0, {Length[lambda]}];
  While[counter ≤ final,
   If[Min[ta] < Min[td],
    {
     For[i = 1, i <= Length[lambda],
      i++, list[[i]] = {ta[[i]], i}];
     list = Sort[list];
     serv = RandomVariate[ExponentialDistribution[mu[[1]]]];
     If[ta[[1]] == list[[1, 1]],
      {
       i = 1;
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = ta[[i]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       If[ lambda[[i]] > 0,
        ta[[i]] = tcurr[[i]] + RandomVariate[
            ExponentialDistribution[lambda[[i]]]], ta[[i]] = Infinity];
      },
```

```
{
    index = list[[1, 2]];
    m = 0;
    n = 0;
    quecus[[index]] = Length[que[[index]]];
    quecus[[1]] = Length[que[[1]]];
    comparserv = serv * mu[[1]] / mu[[index]];
    If[Length[que[[index]]] > 0,
     For[i = 1, i <= Length[que[[index]]],
       i++,
       If[que[[index, m]] ≤ comparserv, m = m + 1];];];
    If[Length[que[[1]]] > 0,
     For[i = 1, i <= Length[que[[1]]],
       i++,
       If[que[[1, n]] ≤ serv, n = n + 1];];];
    If[Length[que[[index]]] > 0,
     delaysubtraction[[index]] = 2 * Sum[que[[index, j]], {j, 1, m}] +
       (2 * quecus[[index]] - 2 * m + 1) * comparserv,
     delaysubtraction[[index]] = comparserv];
    If[Length[que[[1]]] > 0,
     delaysubtraction[[1]] =
      2 * Sum[que[[1, j]], {j, 1, n}] + (2 * quecus[[1]] - 2 * n + 1) * serv,
     delaysubtraction[[1]] = serv];
    If[delaysubtraction[[index]] <= delaysubtraction[[1]], serv = comparserv];
    i = DynamicDispatcher1[delaysubtraction, index];
    tprev[[i]] = tcurr[[i]];
    tcurr[[i]] = list[[1, 1]];
    xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
    ta[[index]] =
     tcurr[[i]] + RandomVariate[ExponentialDistribution[lambda[[index]]]]
   }
   ];
 If[Length[que[[i]]] > 0,
   que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]])];
 que[[i]] = Append[que[[i]], serv];
 que[[i]] = Sort[que[[i]]];
 x[[i]] = x[[i]] + 1;
 counter = counter + 1;
 td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]]
 },
```

```
{
       For[i = 1, i <= Length[lambda],
        i++, listdepart[[i]] = {td[[i]], i}];
       listdepart = Sort[listdepart];
       i = listdepart[[1, 2]];
       tprev[[i]] = tcurr[[i]];
       tcurr[[i]] = listdepart[[1, 1]];
       xx[[i]] = xx[[i]] + (tcurr[[i]] - tprev[[i]]) * x[[i]];
       x[[i]] = x[[i]] - 1;
       If[Length[que[[i]]] > 0,
        que[[i]] = que[[i]] - ((tcurr[[i]] - tprev[[i]]) / Length[que[[i]]]);
        que[[i]] = Delete[que[[i]], 1];
       ];
        If[x[[i]] > 0, td[[i]] = tcurr[[i]] + Length[que[[i]]] * que[[i, 1]],
         td[[i]] = Infinity];
      }
      ]];
  For[i = 1, i <= Length[lambda],
    i++,
    xx[[i]] = xx[[i]] / tcurr[[i]];
    xtotal = xtotal + xx[[i]]
   ];
  xtotal

  ];
DynamicDispatcher1[delaysubtraction_, index_] :=
 Module[{a},
  If[delaysubtraction[[index]] <= delaysubtraction[[1]], Return[index],
    Return[1]
   ];
 ]
```